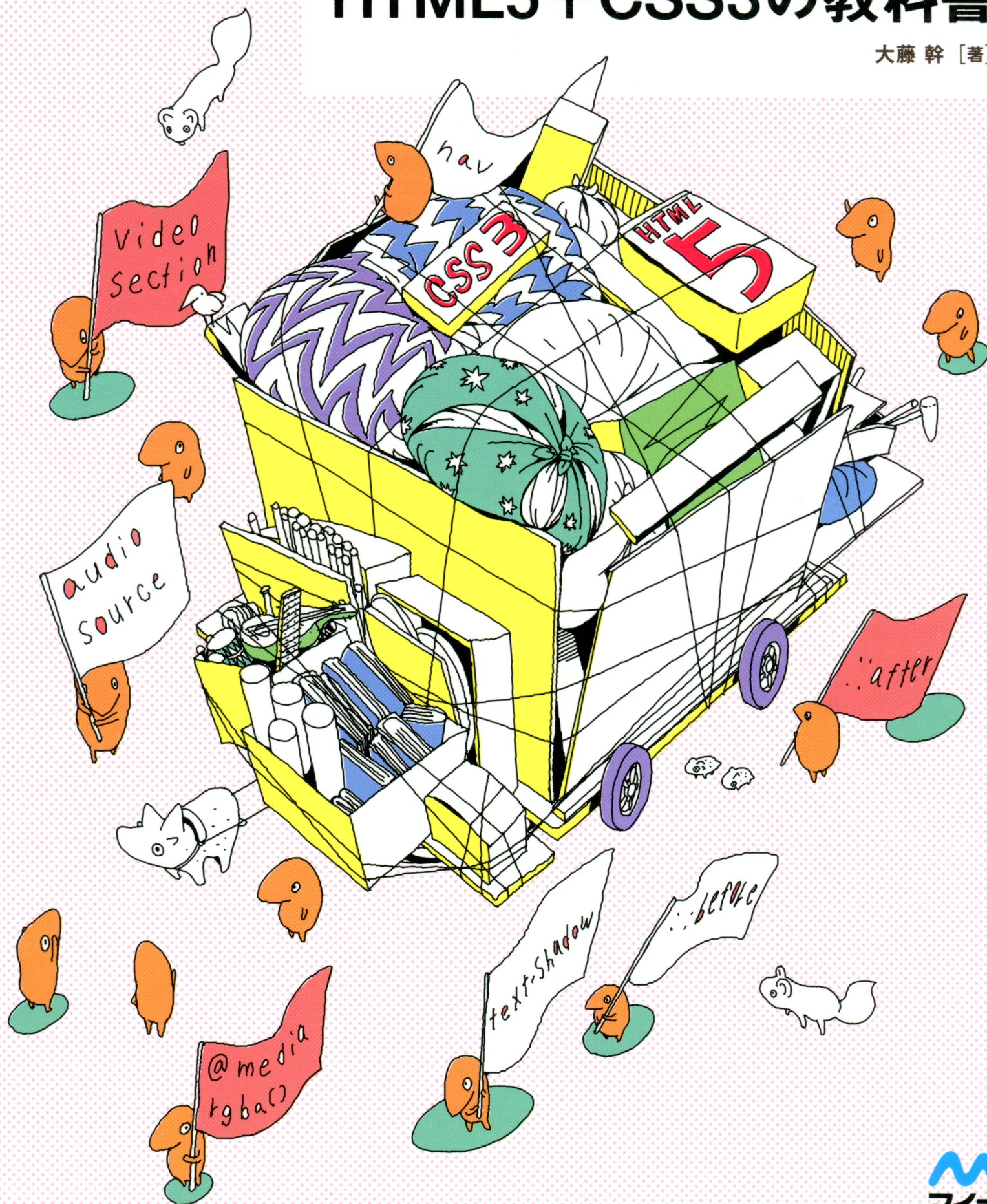
よくわかる HTML5＋CSS3の教科書

大藤 幹［著］



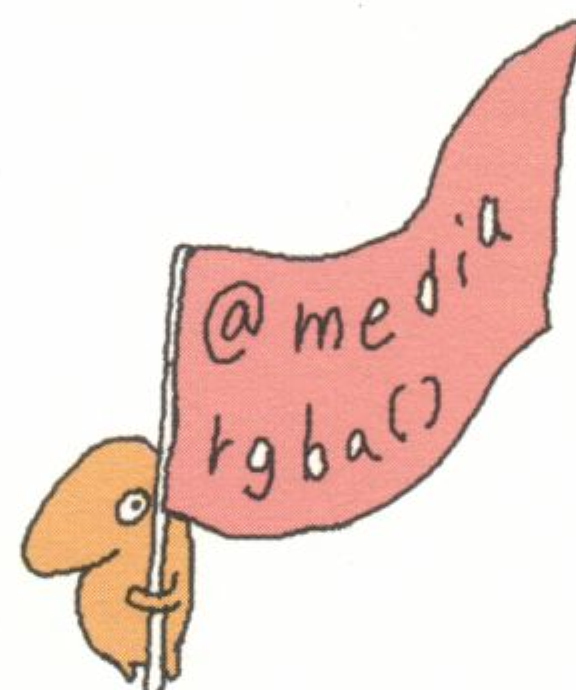
マイナビ

よくわかるHTML5＋CSS3の教科書

CONTENTS

よくわかる
HTML5＋CSS3の教科書

大藤 幹［著］



マイナビ

■本書のサンプルファイルについて

本書のなかで使用されているサンプルファイルは以下のURLからダウンロードできます。

http://book.mynavi.jp/support/pc/4348/

●サンプルファイルのダウンロードにはインターネット環境が必要です。

●サンプルファイルはすべてお客様自身の責任においてご利用ください。サンプルファイルおよび動画を使用した結果で発生したいかなる損害や損失、その他いかなる事態についても、弊社および著作権者は一切その責任を負いません。

●サンプルファイルおよび動画ファイルに含まれるデータやプログラム、ファイルはすべて著作物であり、著作権はそれぞれの著作者にあります。本書籍購入者が学習用として個人で閲覧する以外の使用は認められませんので、ご注意ください。営利目的・個人使用にかかわらず、データの複製や再配布を禁じます。

注意

●本書での説明は、Mac OS X、Google Chrome（一部その他のブラウザ）で行っています。
環境により表示が異なる場合がありますのでご注意ください。

●本書制作時（2012年7月）、HTML5とCSS3の仕様は勧告に至っていない状態であり、執筆以降に変更される可能性があります。

●本書に登場するソフトウェアやURLの情報は本書初版第1刷時点（2012年7月）でのものです。
執筆以降に変更される可能性があります。

●本書の制作にあたっては正確な記述につとめましたが、著者や出版社のいずれも、本書の内容に関して何らかの保証をするものではなく、内容に関するいかなる運用結果についても一切の責任を負いません。あらかじめご了承ください。

まえがき

本書は、これから新しくHTMLとCSSを学びはじめる人向けの入門書です。いま現在の、このタイミングで学びはじめるわけですから、最新バージョンのHTML5とCSS3をベースとした総合的な内容となっています。

この本の企画を進めるにあたっては、クリアすべき課題が2つありました。ひとつは、全体のボリュームの問題です。普通にHTML5とCSS3の両方を解説すると、500ページは軽く越えてしまうことが予想されました。しかし、初心者向けの本を、見ただけで「ちょっと無理」と感じてしまうほど分厚くするわけにはいきません。もうひとつは、構成上の流れの問題です。普通に考えられる順序で解説すると、先にHTMLを学習してからそれを前提としてCSSを学ぶ、という流れになり、前半はタグの意味を覚えることが中心の退屈な内容になってしまうという点です。学習を開始した早々、眠くなるような説明が延々と続いたのでは前向きに頑張る気持ちも失せてしまうかもしれません。

ひとつめの課題については、「現時点でほとんど使われていないもの、仕様が不安定なものについては掲載しない」という方針にすることでクリアしました。ご存知の方も多いかと思いますが、HTML5とCSS3はその大部分がまだ未完成であり、現在でも少しずつ変更されています。まだ正式に決まってもいない内容を入門書で事細かに解説することは、あまり意味のあることだとは思えません。入門者が覚えるべきことは、「将来そう決まるかもしれない仕様書の内容」ではなく、「今現在の実践的な制作のときに必要となる知識」であるからです。

もうひとつの課題に関しては、はじめは仕方がないと考えていました。CSSはHTMLに対して適用するものであり、どうしても先にHTMLを覚える必要があるからです。しかしあるとき、べつにHTMLのすべてを覚えてからCSSに進む必要もないと気づき、執筆の途中で構成全体を作り直しました。HTMLとCSSを少しずつ同時進行させることにしたのです。このような進め方にすると、覚えた内容がすぐにブラウザでの表示結果としてあらわれるため、少しずつできるようになっていく感覚を積み重ねながら学習を進めていくことができます。ただしその分、各段階ではHTMLもCSSも限定された範囲までしか使えないことになるため、新しい構成を考える作業はまるでパズルでも解くかのような複雑な作業となることが予想されました。しかし今回はあえてそれに挑戦してみることにしたのです。その甲斐もあって、HTMLとCSSが同時進行で学べる新しいタイプの教科書が完成しました。

本書を、これからHTMLとCSSを覚える人が“楽しみながら学ぶ”ための入門書としてご活用いただけましたら幸いです。

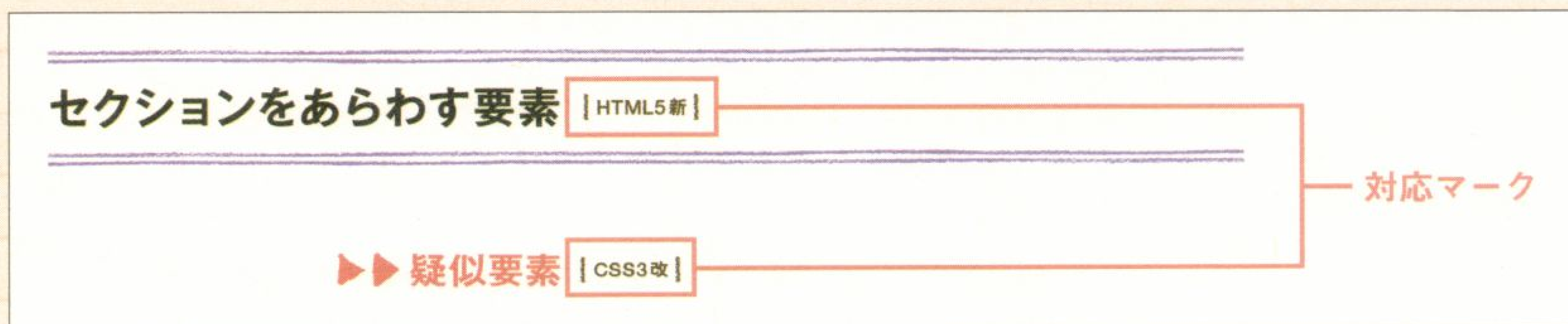
2012年7月

大藤 幹

本書の読み方

対応マークについて

本書のChapter 4以降では、説明項目の一部の見出しにHTMLやCSSのバージョンを示す対応マークが付いています。



対応マークの示す意味は以下の通りです。

HTML5新 …… HTML5で新しく登場した項目、または、大きい変更があった項目です。

HTML5改 …… HTML4.01/XHTML1.0から変更が加えられた項目、または一部に新しい機能が加わった項目です。

CSS3新 ……… CSS3で新しく登場した項目、または、大きい変更があった項目です。

CSS3改 ……… CSS2.1から変更が加えられた項目、または一部に新しい機能が加わった項目です。

HTMLやCSSのバージョンとは特に関係のない内容であるか、概要を説明しているためバージョン対応で考えることがそぐわない内容、または、以前のバージョンとほぼ同じ内容である場合は対応マークは付いていません。

ソースコードについて

本書には、ダウンロード可能なサンプルファイルが用意されています。

ダウンロードサイト
http://book.mynavi.jp/support/pc/4348/

対応するサンプルファイルがある場合は、紙面にパスの表記があります。勉強の参考にしてみてください。

みの1行あいているところ(<body>と</body>の間)にペーストしてください。

sample/chapter-02/lecture-2-2/03.html — サンプルファイルのパス

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
```

※本書に掲載されているソースコードの左側にある行番号は、ソースコードの一部を抜粋して掲載している場合でも常に1からの連番となっています。サンプルファイルの行番号とも一致していない場合がありますので注意してください。

CONTENTS

CHAPTER 1

はじめる準備

Chapter 1では、インターネットの概要とサーバーの役割、Webページを制作して公開する際に必要となるソフトウェアなどの基礎事項について説明します。さっと流し読みしてみて、知っている内容だと思った方は読み飛ばしてもOKです。

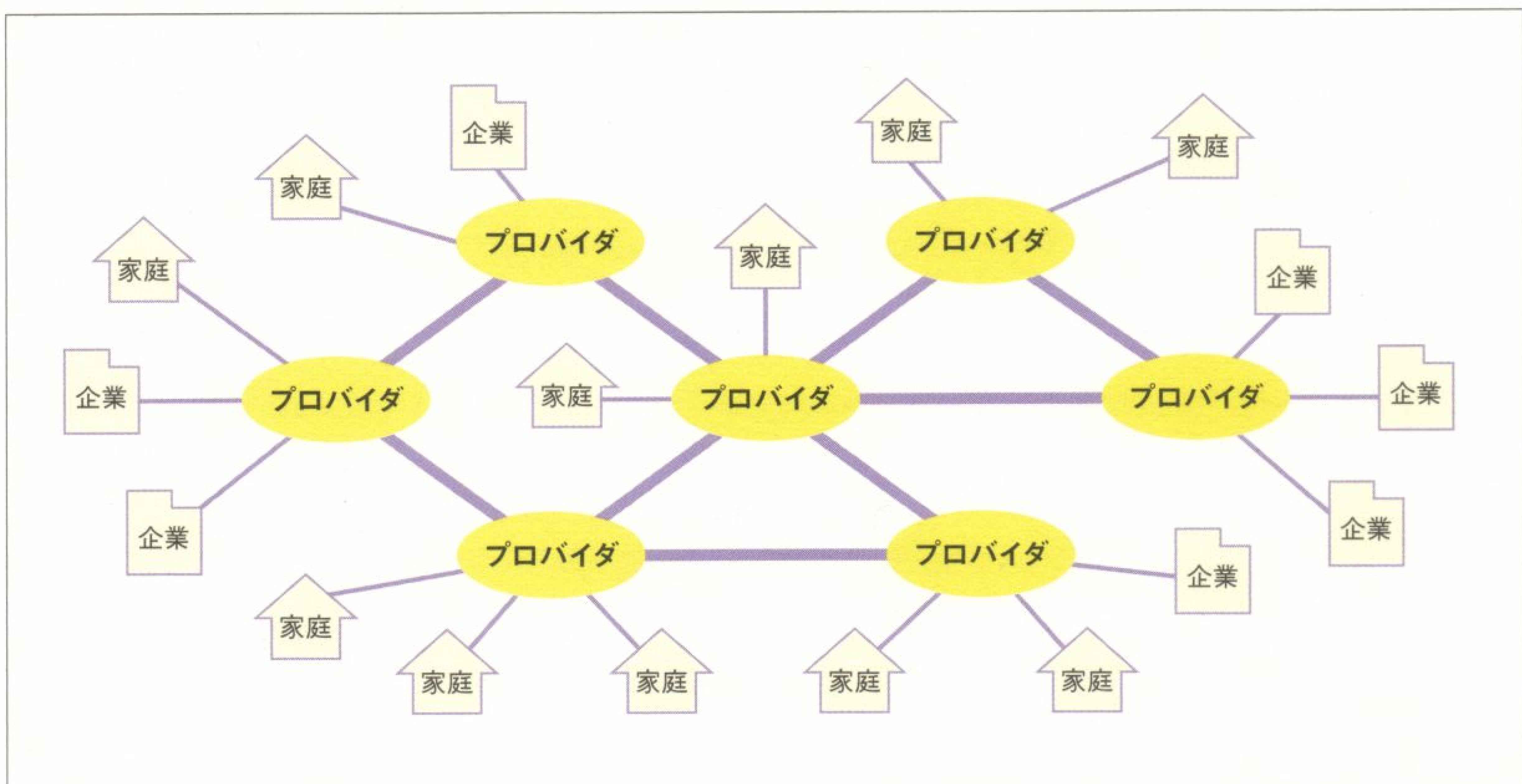
LECTURE 1-1

インターネットとサーバーについて

本書を読むとWebページが作れるようにはなりますが、自分のパソコン上でWebページを完成させたらそれが自動的にインターネットで公開されて誰でも見られるようになる、というわけではありません(もしそうだとしたら、自分のパソコンのファイルがすべてインターネット上で公開されていることになります)。ここではまず、インターネットのおおまかな仕組みと、サーバーの役割について確認しておきましょう。

▶▶ インターネットの仕組み

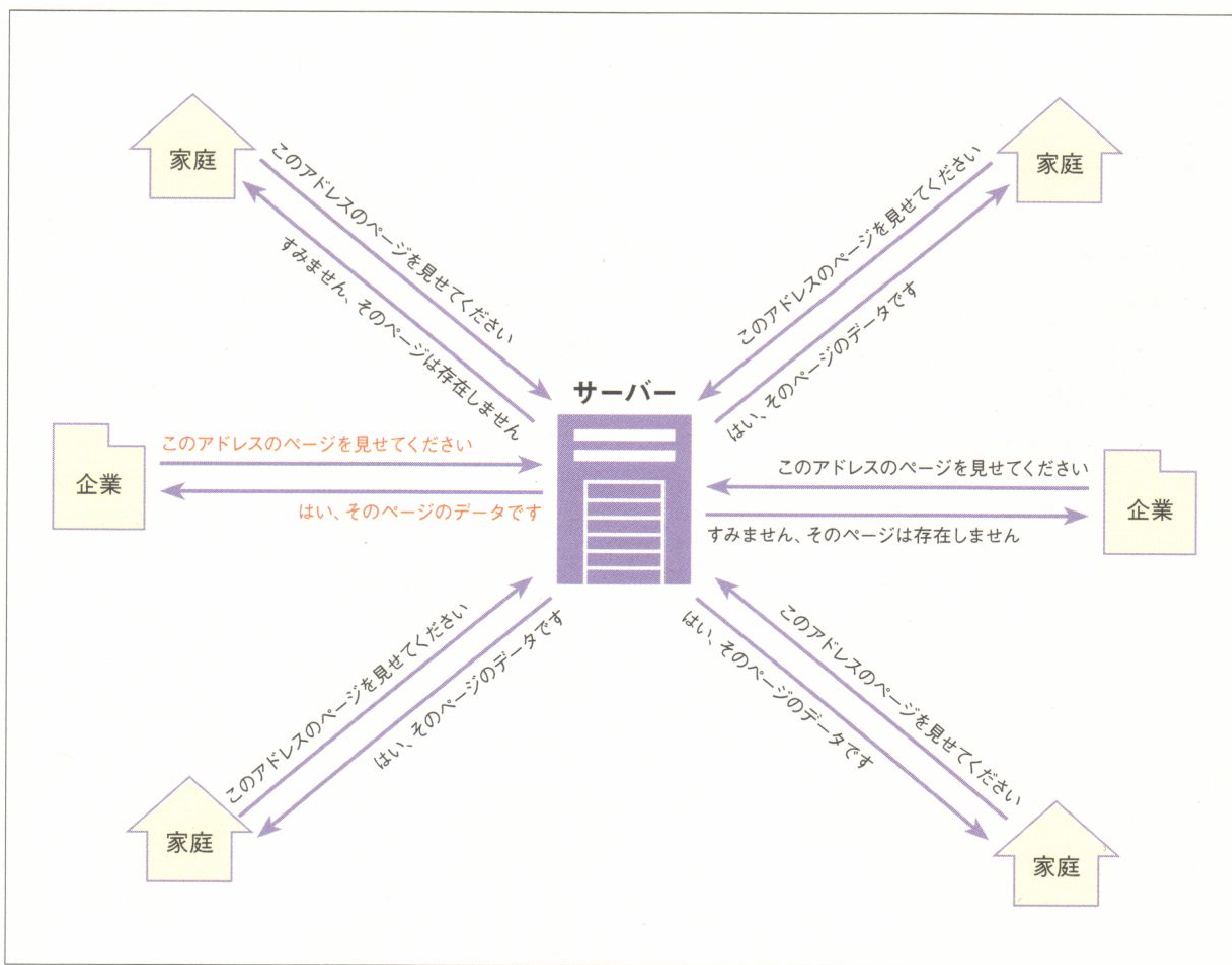
インターネットとは、簡単に言えば「世界中にある個別のネットワークを一定の決まり(TCP/IP)に従って結びつけている世界規模のネットワーク」のことです。家庭や企業からインターネットを利用するためには、すでにインターネットに接続されているプロバイダ(接続業者)と契約して、そこから接続することになります(接続方法には光ファイバーやケーブルテレビの回線、ADSLなど色々あります)。



インターネットのイメージ図

▶▶「インターネットにWebページを公開する」とは

私たちが普段インターネット上のWebページを見ているとき、そのネットワーク上では「このアドレスのページを見せてください」「はいどうぞこれがそのデータです」といったやりとりが何度も行われています。そのようなユーザー（ブラウザ）とのやりとりを行い、ユーザーからの要求に応えてデータを渡すソフトウェアのことをサーバー・ソフトウェアと呼んでいます（そしてサーバー・ソフトウェアが動作しているコンピュータのことをサーバーと言います）。つまり、インターネット上でWebページを公開するには、サーバー・ソフトウェアが動作しているコンピュータにデータを入れるか、Webページのデータが入っているパソコンでサーバー・ソフトウェアを動かす必要があるということです。



サーバーの役割のイメージ図

▶▶ Webページを入れる場所に注意

ただし、サーバー・ソフトウェアが動いていればどのファイルでも公開されてしまうのかといえば、そうではありません。公開するのはサーバー・ソフトウェアで設定している特定のフォルダ内に限定されます。それはもちろん、セキュリティ上の問題があるからです。したがって、インターネット上で公開するためのWebページが完成したら、サーバーが動いているコンピュータ上の特定のフォルダの中に入れることによって、そのデータは初めてインターネット上で公開されることになります。多くの場合、プロバイダと契約すると、プロバイダのサーバーにそのような公開用のフォルダが用意されていて、契約時にどこにデータを入れればよいのか案内があったはずです。また、一般にプロバイダが用意しているサーバーの容量は少なめでさまざまな制限もあるため、接続用のプロバイダのほかに自由度の高いレンタルサーバー(これは接続業者ではなくWebページを置くスペースを提供するサービス)を借りる場合もあるかと思います。その場合は、レンタルサーバー内の指定されたフォルダにデータを入れることになります。

このようにすることで、制作したWebページは初めてインターネット上で公開され、誰でも自由に閲覧できるようになり、Googleなどの検索エンジンにも登録されて検索結果として表示されるようになります。制作したWebページをサーバー内の特定のフォルダに入れるためのソフトウェアについては、次の「Lecture 1-2　本書で使用するソフトウェアについて」で説明します。

LECTURE 1-2

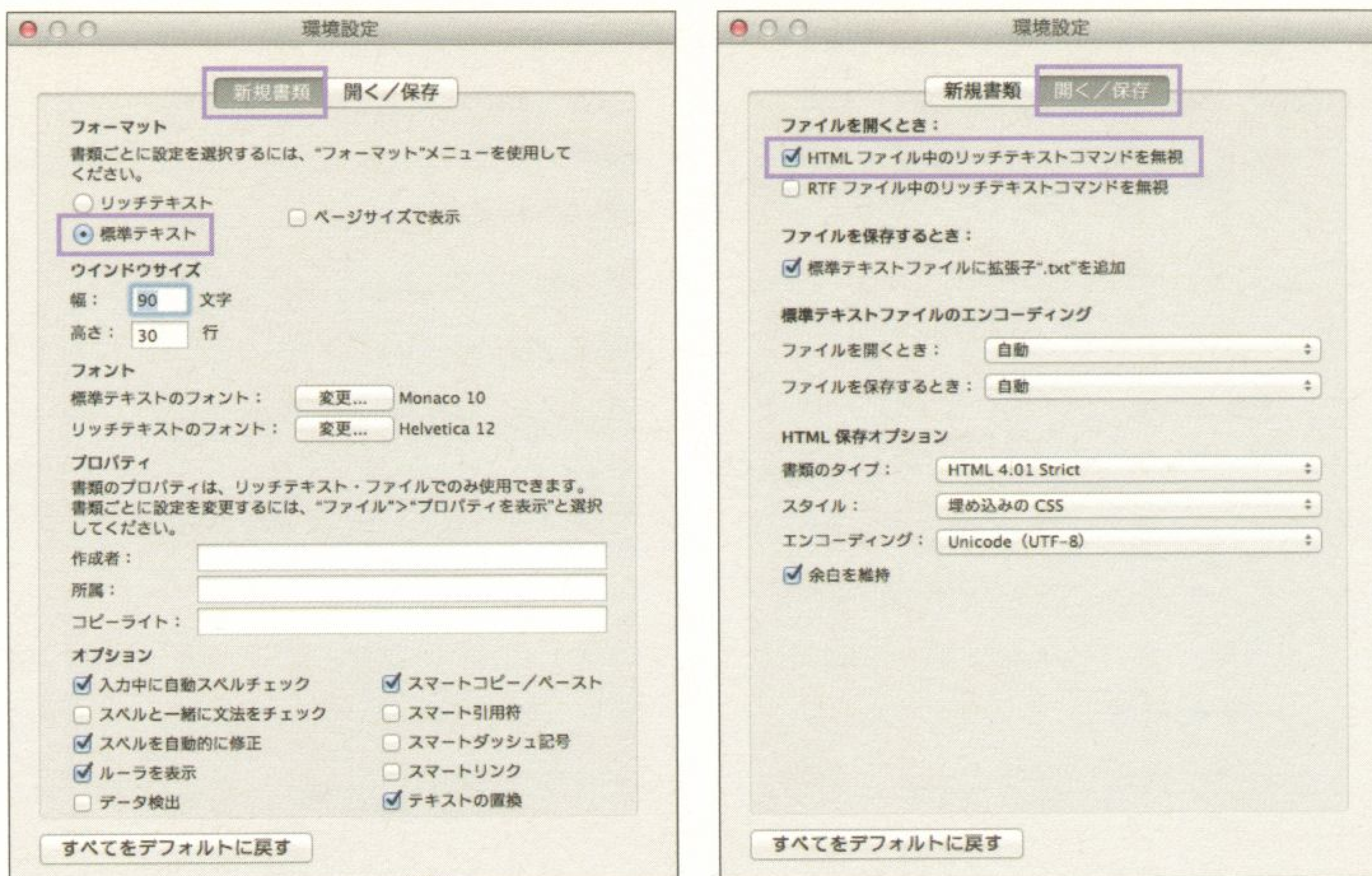
本書で使用するソフトウェアについて

Webページを制作するには専用のソフトウェアが必要だと思っている人もいるかもしれませんが、決してそんなことはありません。なぜなら、HTMLもCSSもJavaScriptも、実際にはただのテキストファイル(文字データだけを含むごく一般的なファイル形式)だからです。つまり、Windowsに付属の「メモ帳」や、Macに付属の「テキストエディット」などがあればWebページを制作することは可能なのです。
ただし、「メモ帳」や「テキストエディット」でもたしかに制作は可能ですが、もっと使い心地がよくて効率的に制作できるソフトウェアも無料で公開されています。また、制作したWebページがさまざまな環境でうまく表示されるかどうかを確認するためには、チェック用のブラウザも用意しておく必要があります。さらに、最終的にできあがったWebページを公開するには、それらをサーバーの特定のフォルダに転送しなければなりません。ここではまず、そのような場面で必要となるソフトウェアを、無料で入手できるものを中心に紹介しておきます。

▶▶ テキストエディタ

テキストエディタ(text editor)とは、文字を入力・編集しテキストファイルとして保存するためのソフトウェアのことです。身近な例としては、Windowsに付属している「メモ帳」やMacに付属している「テキストエディット」が挙げられます。

本書で学ぶHTML5もCSS3も、ファイルの形式としてはテキストファイルであるため、Windowsなら「メモ帳」、Macなら「テキストエディット」で問題なく作成できます。特にHTML5やCSS3を学ぶためにちょっとしたサンプルを作ったりするだけなら、それらで十分であるとも言えます。「メモ帳」なら特に設定などを変更することなくそのまま使えますし、「テキストエディット」でも「環境設定」でフォーマットを「標準テキスト」にして、「開く／保存」のページにある「HTMLファイル中のリッチテキストコマンドを無視」にチェックを入れるだけで使用できます(次ページの図を参照)。



テキストエディットの「環境設定」画面。HTML5やCSS3を編集する場合は、フォーマットを「標準テキスト」にし、「開く／保存」のページにある「HTMLファイル中のリッチテキストコマンドを無視」にチェックを入れる

しかし、HTML5とCSS3を少しでも早く効率的に学習したいと考えているのであれば、早い段階のうちに自分に合ったテキストエディタを用意しておいた方が良いでしょう。一般的なテキストエディタには、それが無料のものであっても、ソースコード[※1]をより読みやすくして入力も容易にする多くの機能が搭載されているからです。たとえば、行数を表示することはもちろん、タグやキーワードなどを色分けして表示する機能や入力を補う機能などのほか、それぞれのテキストエディタには固有の便利な機能が組み込まれています。
テキストエディタを使うのが初めてという方は、はじめは以下のような無料のものを試してみて、必要に応じて有料のものを使用すると良いでしょう。

Windows

・MKEditor for Windows
http://www.mk-square.com/

・TeraPad
http://www5f.biglobe.ne.jp/~t-susumu/library/tpad.html

・サクラエディタ
http://sakura-editor.sourceforge.net/

※1：HTML（マークアップ言語）やCSS（スタイルシート言語）、プログラミング言語などの言語による命令・指示などを記述したテキストをソースコード(source code)と言います。省略して「ソース」と呼ばれることもあります。

Mac

・mi

http://www.mimikaki.net/

・CotEditor

http://sourceforge.jp/projects/coteditor/

高機能で有料のテキストエディタとしては、Windowsでは「秀丸エディタ」、Macでは「Jedit X」が有名です。ちなみに本書の原稿はその「Jedit X」で書かれています。

ブラウザ

みなさんは、いつも自分のお気に入りのブラウザ(Webブラウザ)を使って、さまざまなWebページを閲覧していることと思います。自分がユーザーとして閲覧しているときにはそれでまったくかまわないのですが、作る側になるとそれだけでは足りません。想定されるユーザーが使用していると思われる各種ブラウザで正しく表示されていることを確認する必要があるためです。

たとえば、本書で学習するHTML5とCSS3を問題のない状態で表示させるためには、Google ChromeやSafari、Firefoxなどのできるだけ新しいバージョンを使用する必要があります。Internet Explorerでは、バージョン9であっても未対応の機能が多すぎて、HTML5とCSS3の多くの機能が確認できないことになります。

その一方で、もし仕事でWebページを制作するのであれば、逆に多くの機能に未対応のInternet Explorerのいくつかのバージョンを用意しておく必要もあります。さまざまな機能に未対応のブラウザでどのように表示されるのかも確認する必要があるからです。ユーザーが多くいると想定されるブラウザでは、たとえブラウザ側に問題があって表示結果が変わってしまうとしても、少なくとも書いてある内容がきちんと読めるようにしておかなければなりません。

本書では、Google Chromeを基本ブラウザとして学習を進めていきます。

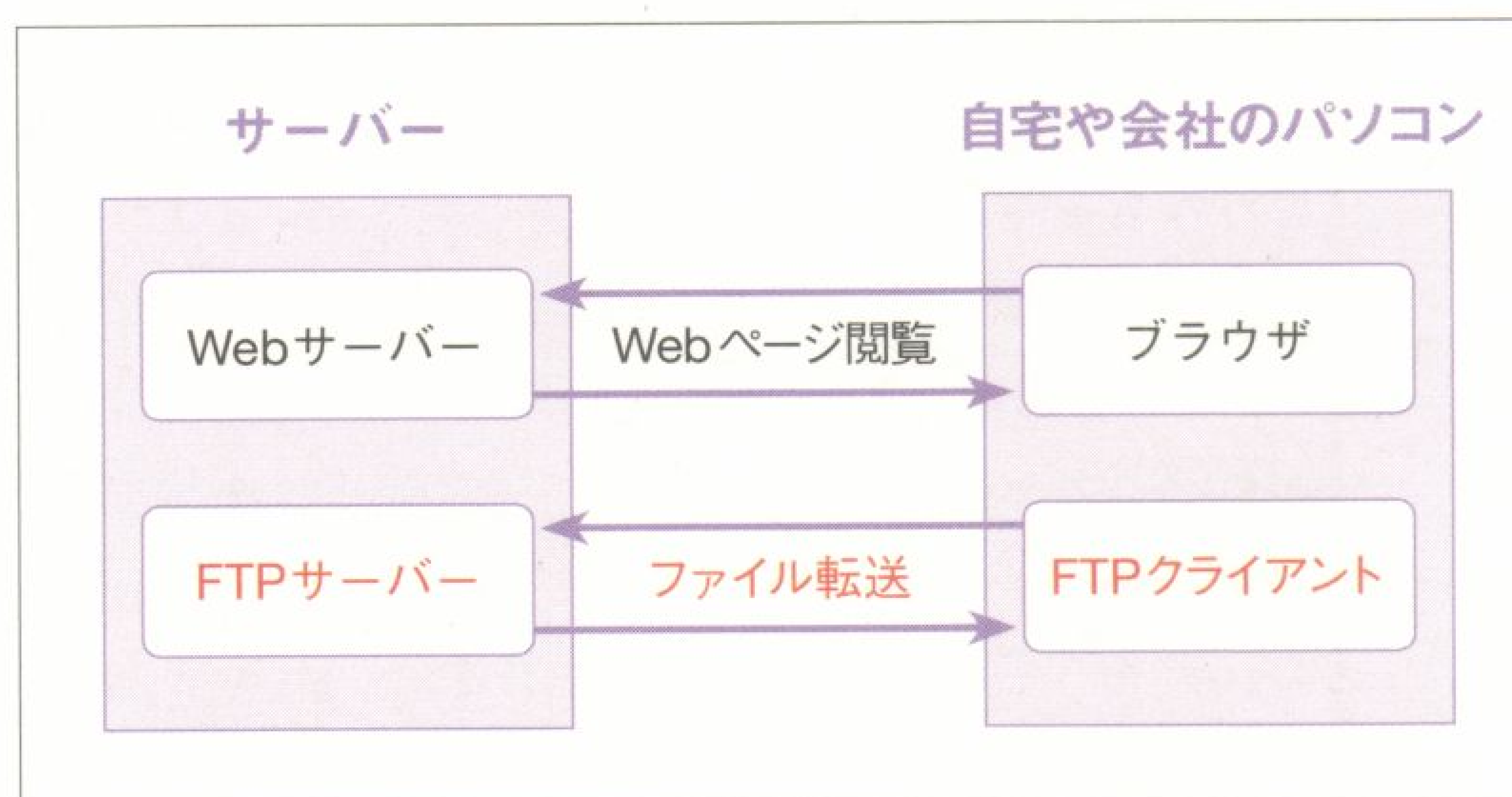


Google Chromeのダウンロードページ。本書では、Google Chromeを基本ブラウザとして学習を進めていく

Safariや Firefoxの最新バージョンでも基本的には同様の表示結果が得られると思われますが、掲載されているサンプルなどのスクリーンショットはGoogle Chromeのものです(必要に応じて他のブラウザのスクリーンショットも掲載しますが、その場合はどのブラウザかを明記してあります)。したがって、本書でHTML5とCSS3を学んでいくだけであれば、ブラウザはGoogle Chromeを用意するだけでOKです。もし、一般公開する予定のWebページを作るのであれば、想定される読者の多くが利用していると思われるブラウザを用意して、表示確認をする必要があります。各種ブラウザをダウンロードするには、ブラウザの名前で検索してみてください。最新バージョンのダウンロード・ページがすぐに見つかります。

▶▶ FTPクライアント

サーバーにはいろいろな種類があります。ブラウザとやりとりを行ってWebページが見られるようにしているサーバーはWebサーバーと呼ばれています。それに対して、ファイルを転送する際のやりとりを行うために用意されているのがFTPサーバーです(FTPは File Transfer Protocol の略です)。Webサーバーとやりとりを行うためにブラウザを使用するように、FTPサーバーとやりとりを行うためにはFTPクライアントというソフトウェアを使用します。



ファイル転送のやりとりを行うのがFTPサーバーとFTPクライアント

FTPクライアントは、できあがったWebページを公開するために、Webページのデータを Webサーバーの特定のフォルダの中に入れるときに使用します。Windows版のFTPクライアントとしては、無料で利用できる以下のソフトウェアが有名です。

・FFFTP
http://sourceforge.jp/projects/ffftp/

また、Firefoxに「FireFTP」というアドオンを追加すると、Windows版でもMac版でもFirefoxにFTPクライアントの機能を追加することができます(こちらも無料で使用できます)。

・FireFTP
https://addons.mozilla.org/ja/firefox/addon/fireftp/

CHAPTER 2

オリエンテーション

Chapter 2では、まずは難しい話は抜きにして、HTMLとCSSとはいったいどんなものなのかということをざっくりと説明し、続けてHTMLとCSSを実際に体験してみます。そこでだいたいの感じをつかんで、HTMLもCSSも意外と簡単なんだな、ということを実感してみてください。

LECTURE 2-1

HTMLの役割、CSSの役割

まずはWebページを構成するHTMLとCSSについて、簡単にその概要と役割を押さえておきましょう。

▶▶HTMLってどんなモノ?

HTMLとは、ごく簡単に言えば、テキストに**タグ**と呼ばれる印を付けて、**それぞれの部分が何であるのか**を示したテキストファイルのことです。それぞれの印は**<ここから見出し>** **<ここまで見出し>**というようなパターンであらかじめ決められていて、HTML5では約100種類あります。HTMLを学ぶということは、その約100種類の**印の意味を知り**、**印の付け方のルールを覚えること**だと言えます。

テキスト

こぶとりじいさん

昔々、あるところに、小太りのお爺さんがいました。
・・・



HTML

<ここから見出し>
こぶとりじいさん
<ここまで見出し>

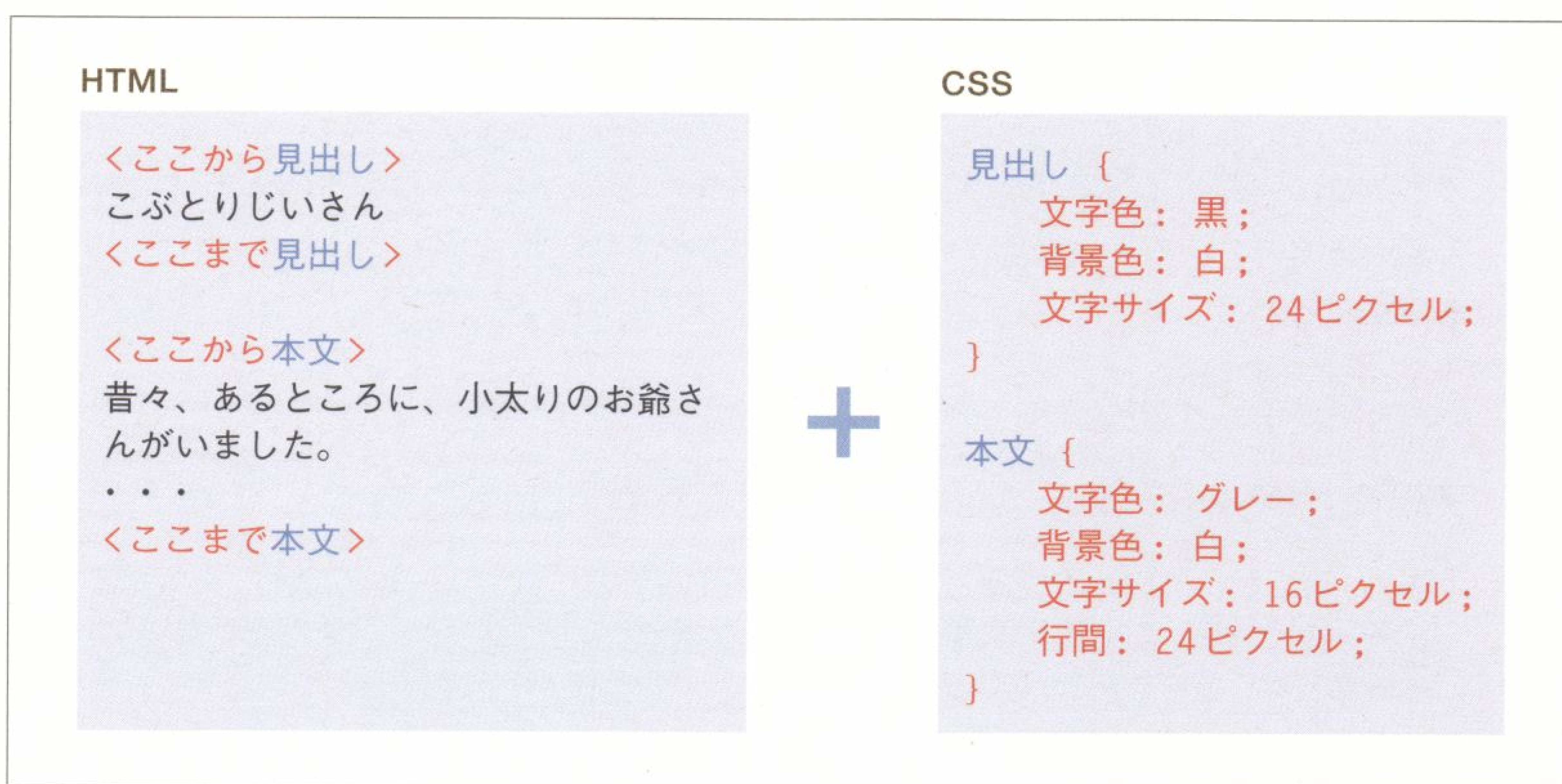
<ここから本文>
昔々、あるところに、小太りのお爺さんがいました。
・・・
<ここまで本文>

HTMLのイメージ。テキストにこのような書式の印を付けて、テキストの各部分が何であるのかを示す

しかし、印の種類が約100種類もあるからといって気を重くする必要はありません。一般的なWebページでふだん使用されているのは、その中のごく一部だからです。すべてを暗記する必要はまったくありませんし、ふだん使うものに関しては使っているうちにすぐに覚えられますので安心してください。このあとに実際にその印を付けてみますが、あっけないほど簡単です。

▶▶ CSSってどんなモノ？

それに対して、HTMLの印によって示された各範囲の表示方法を指定するのがCSSです。たとえば、HTMLで見出しの印をつけた範囲の「文字色」「背景色」「文字サイズ」などを指定できます。その「文字色」や「背景色」のように表示方法として指定できる種類は、HTML5の印の種類よりもかなり多く、ぜんぶだと軽く150以上はあるでしょう（まだすべての仕様が確定しているわけではありませんので最終的にどのくらいの数になるかは現時点では不明です）。ただし、CSSに関してはまだブラウザがサポートしていない機能も多くあり、現実的に使用できるのはだいたい100種類くらいであるとも言えます。



CSSのイメージ。HTMLで示された各部分の表示方法を指定する

HTMLは基本的には印をつけるだけですが、CSSは「どの印をつけた範囲」の「何」を「どう表示させる」といった指定をする必要がありますのでHTMLよりは少々複雑です。しかし、結局は「どう指定すればどう表示させられるのか」ということを覚えるだけですので、プログラミング言語のように難しいわけではありません。CSSもこのあとに実際に使ってみますが、指定方法のパターンさえ覚えれば意外に簡単であることが分かるはずです。

LECTURE 2-2

HTMLのタグをつけてみよう！

いよいよここからは、手を動かしてタグ付けをやってみましょう。使用するサンプルファイルはダウンロードファイルに含まれています。

▶▶ テキストを「大見出し」と「段落」に分ける

それでは、実際にテキストに印をつけてHTMLにしてみましょう。HTMLでは印のことをタグと言いますので、以降本書ではタグという用語を使用します。タグをつけるのは、以下のテキストです。

sample/chapter-02/lecture-2-2/01.txt

```
かちかち山

昔々、ある会社に山田という名前の若い社長さんがおりました。社長仲間のあいだでは「やま」
と呼ばれていました。山田社長には、ことあるごとに「Win-Winの考え方が大切なんです」と
語りだすクセがありました。
```

このテキストにタグをつける

さて、このテキストを読んでみると、どうやらタイトルと本文で構成されているらしいことが分かります。1つのページ内の構成要素として考えると、大見出しと最初の段落であるとも言えそうです。ここではそれを後者であると想定して、まずは日本語で＜ここから大見出し＞＜ここまで大見出し＞＜ここから段落＞＜ここまで段落＞というタグをつけてみることにしましょう。

```
＜ここから大見出し＞
かちかち山
＜ここまで大見出し＞

＜ここから段落＞
昔々、ある会社に山田という名前の若い社長さんがおりました。社長仲間のあいだでは「やま」
と呼ばれていました。山田社長には、ことあるごとに「Win-Winの考え方が大切なんです」と
語りだすクセがありました。
＜ここまで段落＞
```

タグをつけてみた状態

タグがついて、ページ内の2つの構成要素が何であるのかが明確に示されました。しかし、タグがつけられたテキストを見てみると、タグに含まれている「ここから」と「ここまで」は無い方がすっきりして見やすいような気もします。そこで、「ここから」はそのままカットしてしまい、「ここまで」は代わりに「/」であらわすことにして、次のように変更します。

```
<大見出し>
かちかち山
</大見出し>

<段落>
昔々、ある会社に山田という名前の若い社長さんがおりました。社長仲間のあいだでは「やま」
と呼ばれていました。山田社長には、ことあるごとに「Win-Winの考え方が大切なんです」と
語りだすクセがありました。
</段落>
```

「ここから」をカットし、「ここまで」を「/」に変更する

▶▶ タグを英語にする

これですっきりとして見やすくなりました。しかし、HTMLのタグは日本語ではなく英語で作られているはずです。そこで、タグ内の日本語部分を英語にしてみましょう。

まず、「見出し」は「heading」にします。大見出しの「大」はそのまま「大」「中」「小」のようにあらわしてもいいのですが、そうすると見出しの種類が「大見出し」「中見出し」「小見出し」の3段階に限定されてしまいます。ここではもっと多くの階層の見出しが表現できるように、大きい方の見出しから順に数字であらわすようにしてみましょう。ここでは、「大見出し」は「heading1」ということにします。
次は「段落」です。「段落」は英語では「paragraph」です。というわけで、タグの日本語部分を英語に変更すると次のようになります。

```
<heading1>
かちかち山
</heading1>

<paragraph>
昔々、ある会社に山田という名前の若い社長さんがおりました。社長仲間のあいだでは「やま」
と呼ばれていました。山田社長には、ことあるごとに「Win-Winの考え方が大切なんです」と
語りだすクセがありました。
</paragraph>
```

タグ内の日本語を英語に変更

▶▶ さらにタグを短くする

やっと英語にはなったのですが、一度はすっきりとして見えていたタグが、英語に変更したとたんに何だか少し見にくくなってしまいました。本来、タグは印なのですから、ごちゃごちゃした長い単語ではなく、短くて記号のようにすっと目に入ってくる方がよさそうです。そんなわけで、ここは思いきって単語の先頭の文字だけを使うように変更してみましょう。

```
<h1>
かちかち山
</h1>

<p>
昔々、ある会社に山田という名前の若い社長さんがおりました。社長仲間のあいだでは「やま」と呼ばれていました。山田社長には、ことあるごとに「Win-Winの考え方が大切なんです」と語りだすクセがありました。
</p>
```

英単語は短くして、単語の１文字目だけを使うことにする

とてもすっきりとして、見やすく、印らしくなりました。そして、最初は日本語で仮につけてみたタグが、この段階で見事にHTMLのタグに変身しています[※2]。ここでは、HTMLのタグの本来の意味を分かりやすく示すために少々まわりくどいタグの付け方をしてみましたが、実際にWebページを制作する場合は、テキストに単純に<h1></h1>や<p></p>を挿入するだけでOKです。

※2：これらは、実際にもっとも頻繁に使用されるHTMLのタグなのですが、すべてのタグがこれらのように先頭の１文字だけを使う省略形になっているわけではありません。HTMLのタグの中には、別の省略方法がとられているものや、まったく省略されていないものもあります。

▶▶ ページ全体の枠組みを作る

ところで、HTMLのタグの付け方としてはこれでOKなのですが、これだけでHTMLが完成したわけではありません。文法的に正しいHTMLファイルにするためには、これらの他にページ全体の枠組みとなるタグが必要なのです。具体的には、次のようなものです。意味不明で少し複雑なものに見えるかもしれませんが、今の段階ではこれらは“決まり文句”のようなものだと考えておいてください。

sample/chapter-02/lecture-2-2/02.html[※3]

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
```

```
<title>サンプル</title>
</head>
<body>

</body>
</html>
```

全体の枠組みとなるタグ

※3：一般的なテキストファイルの拡張子は「.txt」ですが、HTMLファイルの場合は「.html」または「.htm」となります(サーバーの設定で変更することも可能です)。

では、この枠組みの中に先程タグをつけたテキストを組み込んで、HTMLファイルとして完成したものにしてみましょう。最初にタグをつけたテキスト全体を選択してコピーし、ページ全体の枠組みの1行あいているところ(<body>と</body>の間)にペーストしてください。

sample/chapter-02/lecture-2-2/03.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>サンプル</title>
</head>
<body>
<h1>
かちかち山
</h1>
<p>
昔々、ある会社に山田という名前の若い社長さんがおりました。社長仲間のあいだでは「やま」と呼ばれていました。山田社長には、ことあるごとに「Win-Winの考え方が大切なんです」と語りだすクセがありました。
</p>
</body>
</html>
```

タグをつけたテキストを、ページ全体の枠組みの中に組み込む

これで文法的にもまったく問題のない完璧なHTMLファイルが完成しました。HTMLファイルを作るということは、このようにテキストの各部分に<h1></h1>や<p></p>のようなタグをつけていく作業が中心となります。

つまり、HTML側でどのようなタグが用意されているのかをざっと覚えておき、テキストを見てそこにふさわしいタグをつけられるようになればOKなのです。誰にでもできる簡単な作業ですので、安心してどんどん進めていきましょう。

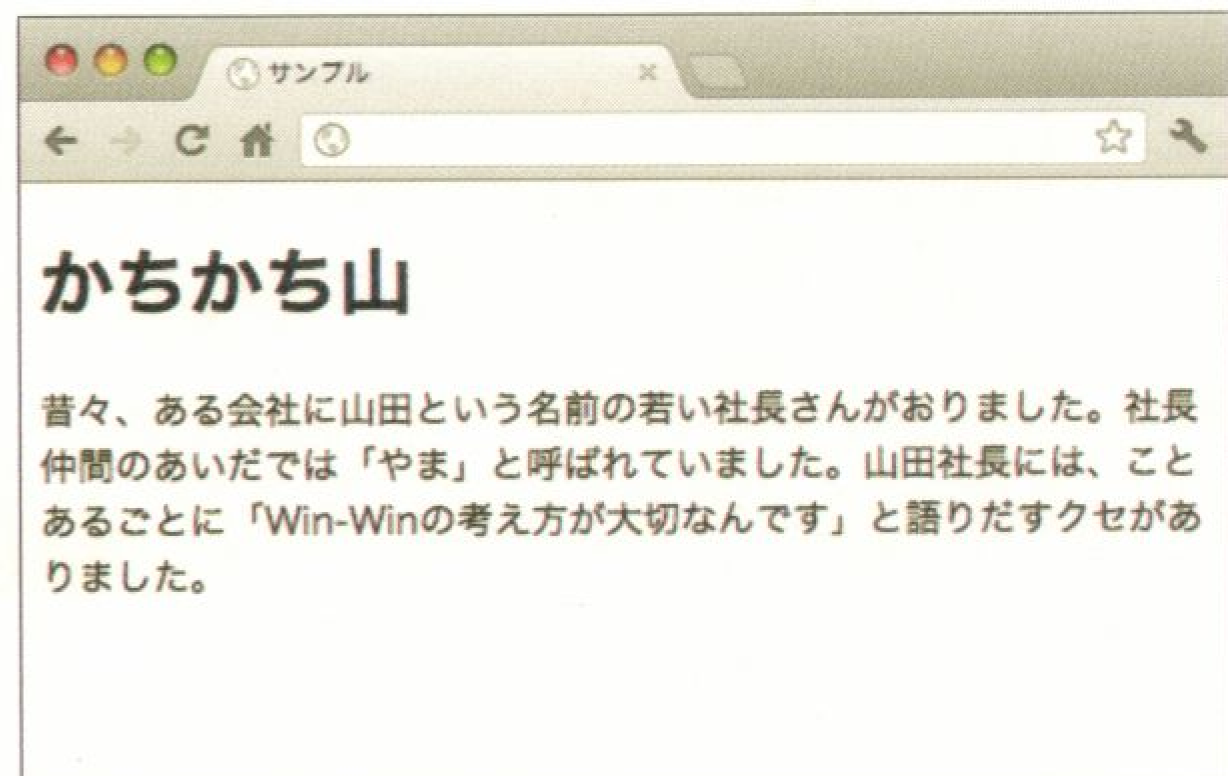
LECTURE 2-3

CSSを使ってみよう！

次はCSSを使ってみましょう。HTMLでタグ付けをした部分に、表示の指定をしてみます。

▶▶ CSSファイルを読み込ませる

さて、文法的に正しいHTMLファイルは完成しましたが、表示方法はまだ一切指定していません。この段階でHTMLを表示させるとどのようになるのかを確認してみましょう。Lecture 2-2で完成させたファイル（sample/chapter-02/lecture-2-2/03.html）をブラウザで開いてください。ほぼ右のように表示されるはずです（ブラウザの設定によって表示結果は異なります）。



表示指定をしていない状態でのHTMLの表示

では、さっそくこれにCSSを適用してみましょう。「lecture-2-3」フォルダの中には、すでにCSSを書き込んだファイル「style.css」を用意してありますので、HTMLの中に「style.css」を読み込む指定を追加します。CSSファイルを読み込ませるには、以下のような指定を追加してください。

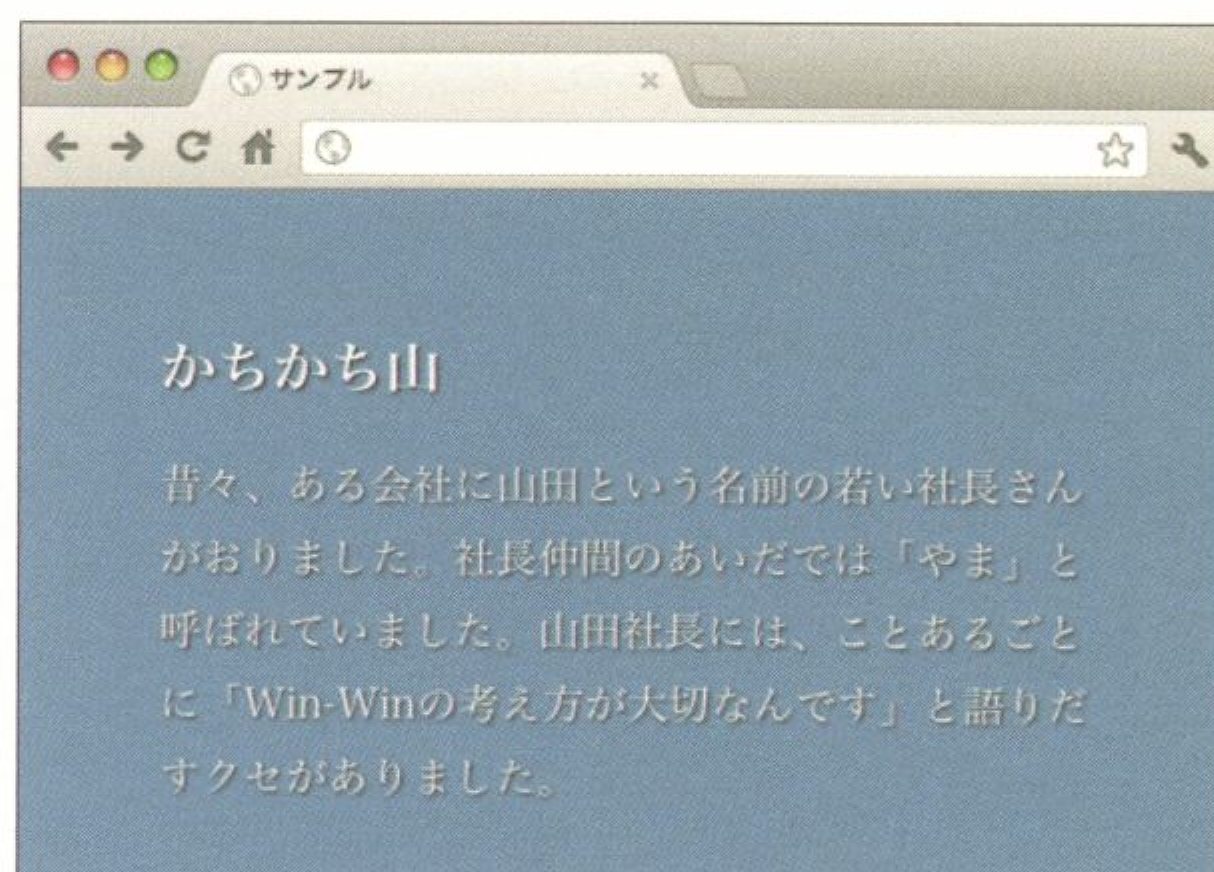
sample/chapter-02/lecture-2-3/index.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>サンプル</title>
<link rel="stylesheet" href="style.css">   ← この行を追加
</head>
<body>
<h1>
かちかち山
</h1>
<p>
昔々、ある会社に山田という名前の若い社長さんがおりました。社長仲間のあいだでは「やま」と呼ばれていました。山田社長には、ことあるごとに「Win-Winの考え方が大切なんです」と語りだすクセがありました。
</p>
```

```
</body>
</html>
```

CSSファイル「style.css」を読み込む指定を追加

「lecture-2-3」フォルダ内の「index.html」は、CSSを読み込む指定をすでに追加してありますので、それをブラウザで表示させてみましょう。
右のように表示されるはずです（表示結果はOSの種類やそのバージョンによって多少の違いがあります）。



CSSファイルを読み込ませたあとの表示結果

▶▶ 読み込ませたCSSの内容を確認する

表示結果ががらりと変化しました。続けて、具体的にどのようなCSSを指定すればこのような表示になるのかを確認してみましょう。
テキストエディタで、「lecture-2-3」フォルダ内の「style.css」を開いてください。

では、ざっくりとですが、この指定内容を説明します。赤で示した部分は、それぞれ**何に対する表示指定**であるのかを示しています。
先頭にある**body**というのは、HTMLの枠組みの中にあった<body>と</body>の範囲のことで、ひとことで言えば**ページ全体**を意味します。**h1**はもちろん<h1>と</h1>で囲った**大見出し**のことで、**p**は<p>と</p>で囲った**段落**のことです。
つまり、body { }の中に書かれているのは**ページ全体に対する表示指定**、h1 { }の中に書かれているのは**大見出しの表示指定**、

sample/chapter-02/lecture-2-3/style.css※4

```
body {
  margin: 60px;
  font-family: serif;
  background: steelblue;
}

h1 {
  color: white;
  font-size: 24px;
  text-shadow: 1px 1px 2px black;
}

p {
  color: white;
  font-size: 18px;
  text-shadow: 1px 1px 2px black;
  line-height: 1.8;
}
```

「style.css」に書かれている全内容

※4：CSSファイルの拡張子は「.css」です（サーバーの設定で変更可能です）。

p { } 中に書かれているのは段落の表示指定、ということです。シンプルで分かりやすい書式です。表示指定もシンプルで、たとえば「color: white;」は文字色を白にする指定です。つまり、「○○○ : □□□ ; 」の書式で「何を : どうする ; 」といったパターンで指定すればよいのです。文字色を赤にしたければ「color: red;」でOKです。
表示指定の「何を」の部分を見るとおおよその予測ができるかと思いますが、bodyに対する指定のmarginはマージン、font-familyはフォントの種類、backgroundは背景の指定となります。h1とpに指定されているfont-sizeはフォントサイズ、text-shadowは文字につける影、line-heightは行間です。

▶▶ CSSを書き換えてみる

CSSがどのようなものであるのか、おおまかなイメージがつかめたと思いますので、「style.css」に書かれている内容を少し書き換えてみて、表示がどのように変化するのかを見てみましょう。
まずはページ全体の背景色を変えてみましょう。現在は「steelblue」という色になっていますが、これを「orange」に書き換えて上書き保存します。

```
body {
  margin: 60px;
  font-family: serif;
  background: orange;
}

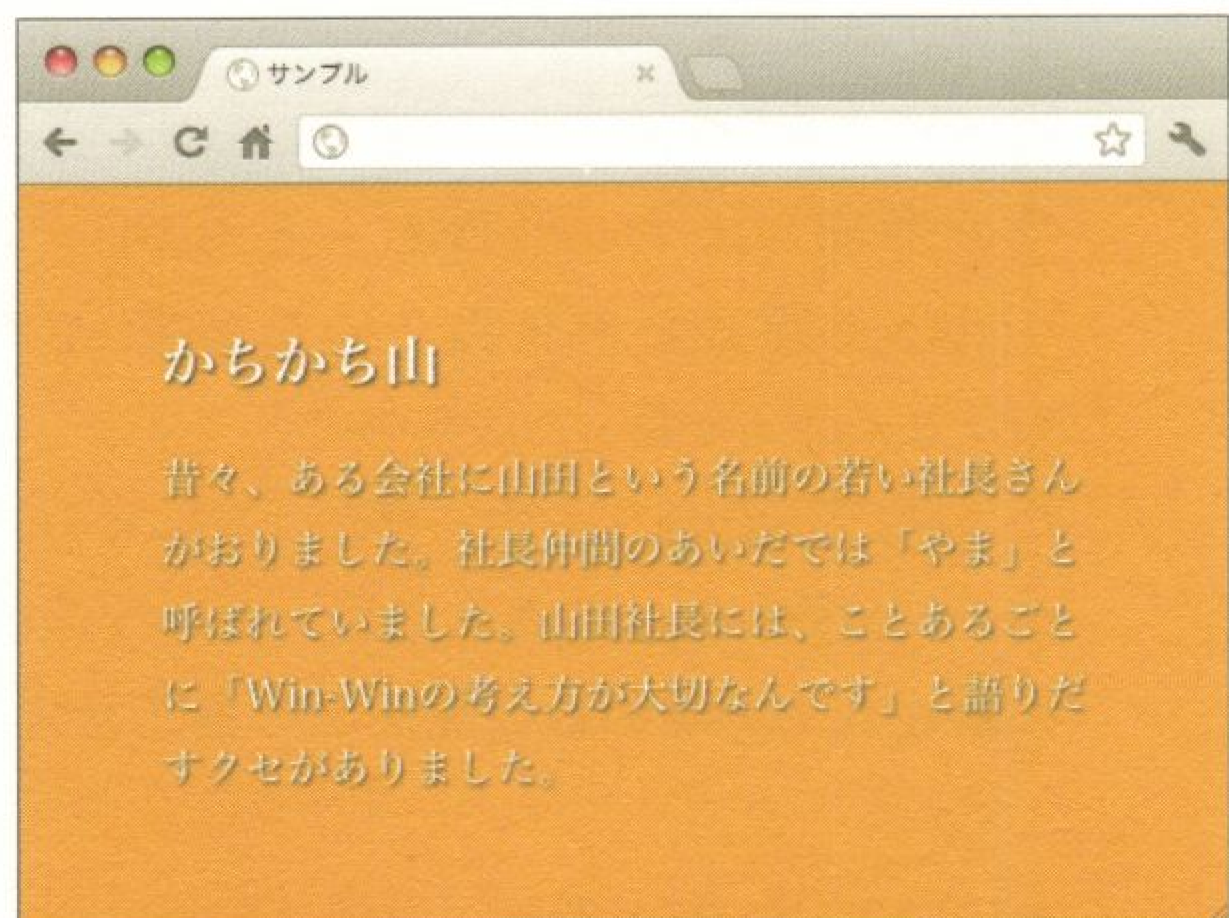
h1 {
  color: white;
  font-size: 24px;
  text-shadow: 1px 1px 2px black;
}

p {
  color: white;
  font-size: 18px;
  text-shadow: 1px 1px 2px black;
  line-height: 1.8;
}
```

steelblueをorangeに変更

ページ全体の背景色を変更する

「index.html」をブラウザで再表示させると、ページ全体の背景色がオレンジ色に変わっています。



ページ全体の背景色がオレンジ色になった

次は大見出しのフォントサイズを変更します。現時点では「24px」、つまり24ピクセルになっていますので、これを「50px」に変更します。ピクセル単位であることを示す「px」の部分はそのままにして、数字だけを24から50に変更すればOKです。このとき、数字とpxのあいだにスペースを入れないようにしてください。書き換えが終了したら、上書き保存します。

```
body {
  margin: 60px;
  font-family: serif;
  background: orange;
}

h1 {
  color: white;
  font-size: 50px;
  text-shadow: 1px 1px 2px black;
}

p {
  color: white;
  font-size: 18px;
  text-shadow: 1px 1px 2px black;
  line-height: 1.8;
}
```

24pxを50pxに変更

h1のフォントサイズを変更する

「index.html」をブラウザで再表示させると、大見出しのフォントサイズが50ピクセルに変わっています。



大見出しの文字サイズが大きくなった

次に、大見出しと段落内のテキストにつけられているテキストの影を消してみましょう。そうするには、右のように text-shadow ではじまる行を削除するだけでOKです。削除したら、上書き保存してください。

```
body {
  margin: 60px;
  font-family: serif;
  background: orange;
}

h1 {
  color: white;
  font-size: 50px;
  text-shadow: 1px 1px 2px black;
}

p {
  color: white;
  font-size: 18px;
  text-shadow: 1px 1px 2px black;
  line-height: 1.8;
}
```

この行を削除する

この行を削除する

h1とpの文字に影をつける指定を行ごと削除する

「index.html」をブラウザで再表示させると、大見出しと段落内のテキストにつけられていた影が消えています。



テキストの影が消えた

最後に、段落内のテキストの行間を変更してみましょう。現時点では「1.8」、つまりフォントサイズの1.8倍に設定されていますので、ここではそれを1.0倍に変更してみます。「1.8」を「1.0」に書き換えて上書き保存してください。

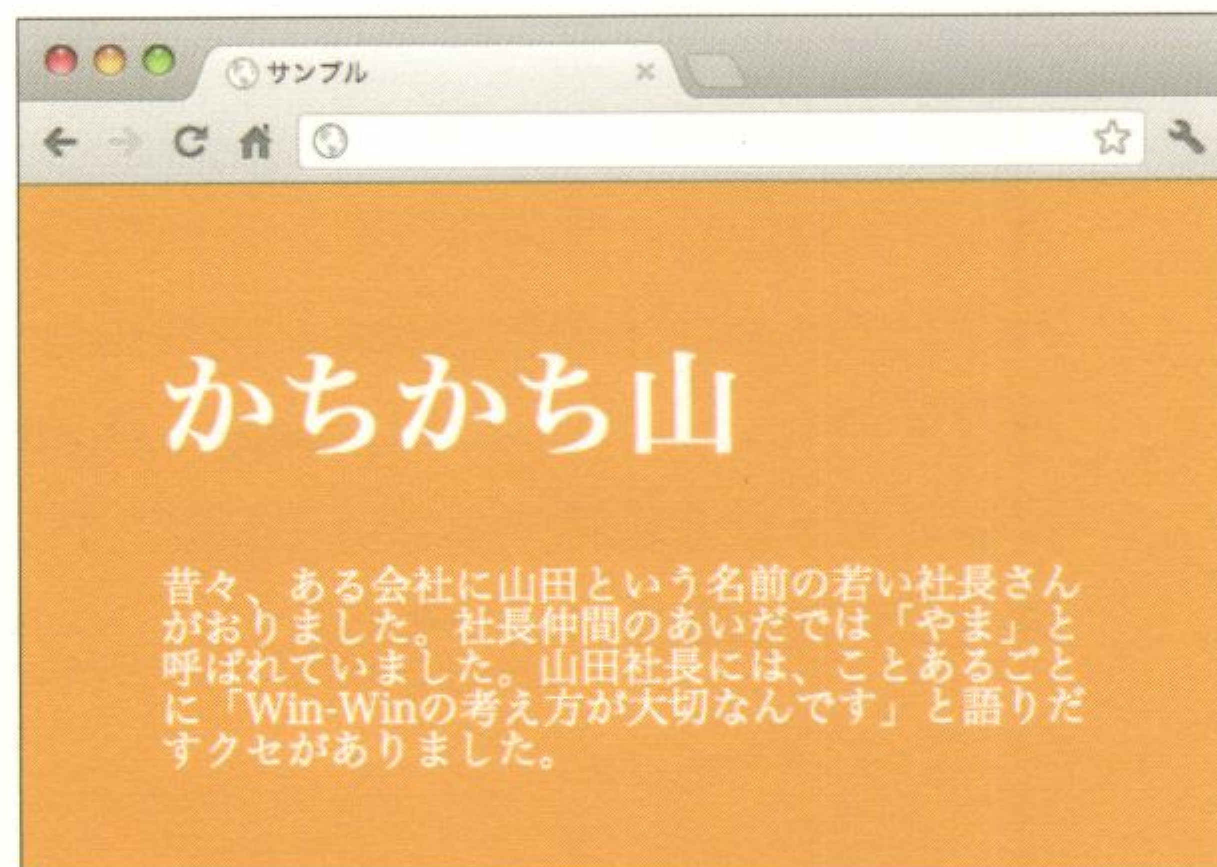
```
body {
  margin: 60px;
  font-family: serif;
  background: orange;
}

h1 {
  color: white;
  font-size: 50px;
}

p {
  color: white;
  font-size: 18px;
  line-height: 1.0;  ← 1.8を1.0に変更
}
```

pの行間を変更する

「index.html」をブラウザで再表示させると、段落内の行間が狭くなっています。



段落の行間が狭くなった

たいへん簡単なサンプルではありましたが、実際にテキストにタグをつけてHTMLにし、それにCSSを適用して表示方法も少し変更してみました。これでHTMLとCSSのイメージは掴むことができたのではないかと思います。あとは細かいことを少しずつ覚えていけばいいだけです。楽しみながらどんどん進めていきましょう！

といいつつ、次の Chapter 3のタイトルを見ると「文法的なカタい話」となっています(書き方の基本的なルールや専門用語などを学習します)。退屈でちょっと眠たくなるかもしれない内容ですが、HTMLとCSSをきちんと覚えるには避けて通ることのできない重要な部分ですので、ここだけはちょっと頑張って覚えてください。

CHAPTER 3

文法的なカタい話

これからHTMLのタグとCSSの表示指定を少しずつ学習していきますが、その前にHTMLとCSSの説明で使われる用語や書式の基本ルールを覚えておきましょう。Chapter 4以降をスムーズに進めていくにはどうしても必要な知識ですので、しっかりと覚えてください。

LECTURE 3-1

HTMLのタグを正しくつける意味

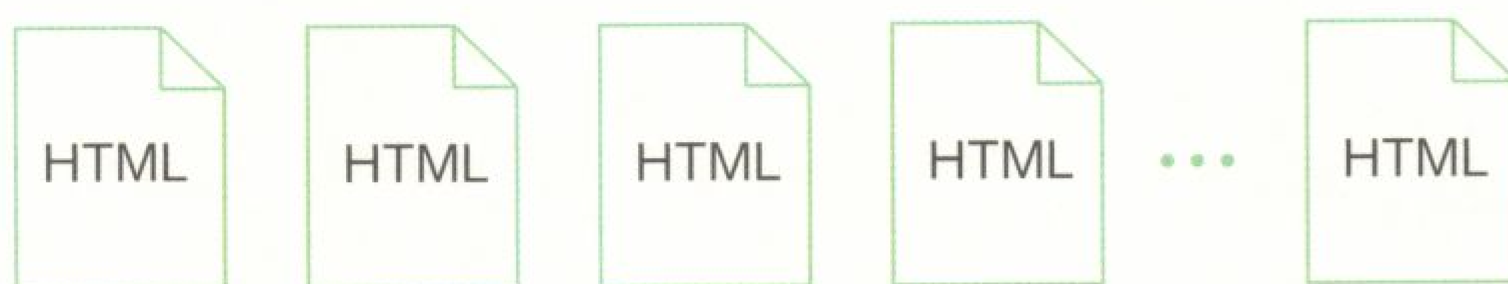
テキストをどのようにタグ付けするかは、実はとても大切なことです。
「正しく」タグ付けすることにどんなメリットがあるのか、知っておきましょう。

▶▶ CSSを独立させるメリット

かつてのWebページは、ワープロと同様の感覚で、どう表示させるかということだけを考えて制作されていました。しかも、CSSは一切使わずに、HTMLの表を作る機能を利用して、HTMLだけで作られていたのです(いわゆるテーブルレイアウトという制作手法です)。

しかし、それでは当然のように無理が生じます。HTMLで表示指定を行うと、ページ内の各部分ごとに表示指定を埋め込んでいくことになりますので、HTMLファイルの容量はかなり大きなものとなります。そして、表示指定がその場所ごとに埋め込まれているということは、たとえば100ページあるサイトの表示を変更しようと思ったら、100ページすべてを修正する必要があることにもなります。また、表示指定はその埋め込まれたもので固定となり、パソコンの画面で閲覧することにしか対応していないページになってしまいます。つまり、容量がムダに大きくて、変更に大きな手間がかかり、閲覧環境を制限してしまう融通のきかないページになってしまうということです。

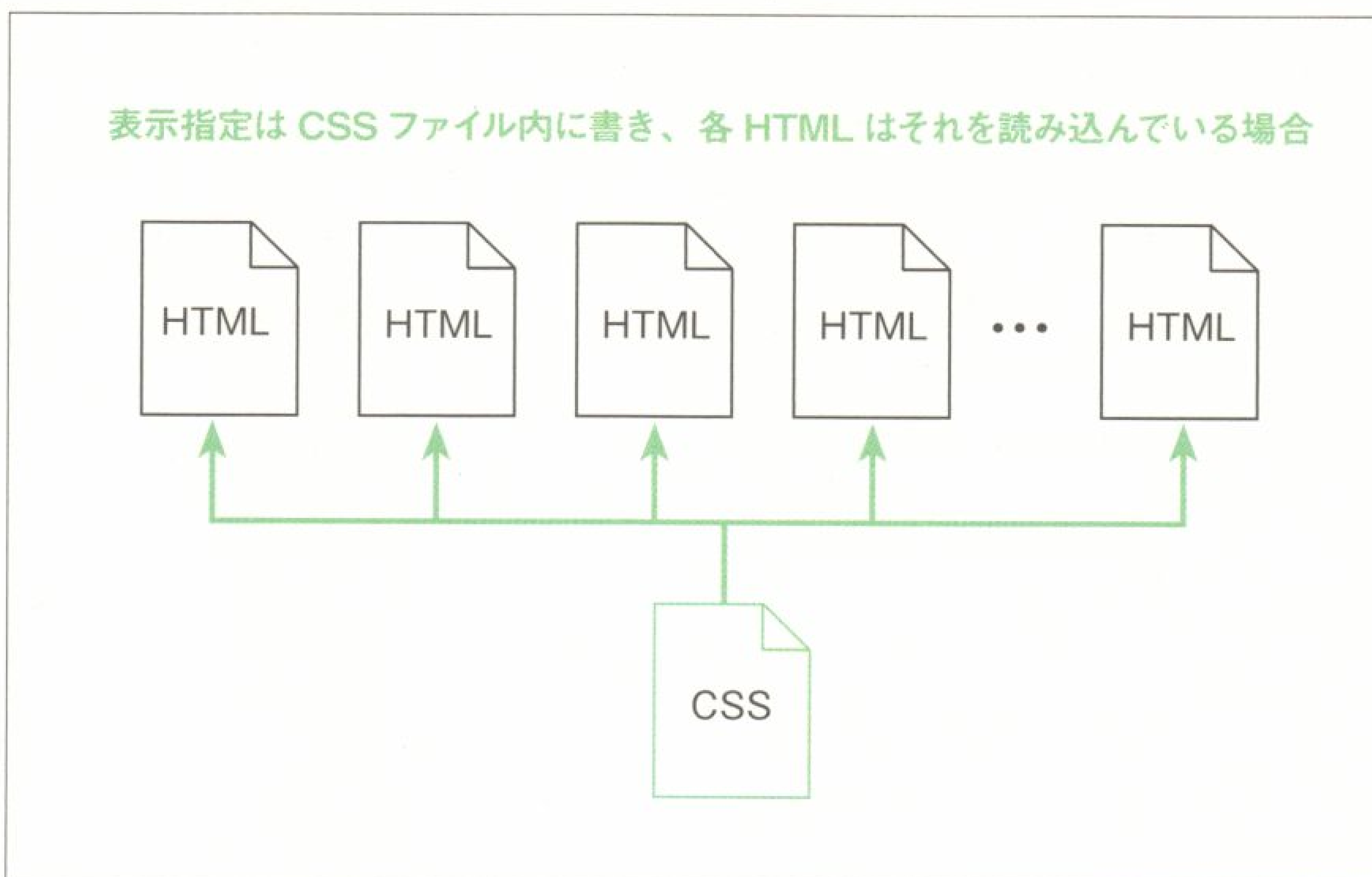
それぞれのHTMLファイル内に表示指定が組み込まれている場合



HTMLで表示指定を行った場合、100ページの表示変更をするなら、100ページすべてを修正する必要がある

そこで、徐々にではありましたが、表示指定にCSSが利用されるようになってきました。表示指定をHTMLから取り除き、独立したCSSファイルにして、それを各HTMLから読み込ませる方式に変更したのです。これによって、HTMLファイルの容量は劇的に少なくなり、100ページの表示変更もHTMLに手を加えることなくCSSを修正するだけで済むようになりました。また、たとえ

ば印刷用の表示指定を別途用意したり、画面サイズ別のレイアウトを用意したりすることも可能になりました。



CSSで表示指定を行えば、100ページの表示変更でもCSSを変更するだけでOK

▶▶HTMLのタグ付けを正しくするメリット

しかし、CSSには大変分かりやすい大きなメリットがあったために使われるようになってはきたものの、HTMLのタグに関してはあまり注意を払わずに作る人が多くいました。具体的に言うと、ページ内の多くの部分に対して<div>～</div>というタグばかりをつけてしまうという制作方法です。これはつまり、HTMLを「ページ内の構成要素として何であるかを示す印」として使うのではなく、「単純に範囲だけを示す印」として使っていることになります。それでもCSSを使用するメリットの多くは得られますが、それではWebページが本来持っているポテンシャルの一部しか発揮できないページになってしまいます。

```
<div id="h001" class="heading">
○○○○○
</div>
<div class="honbun">
○○○○○○○○○○○○○○○○○○○○○○
</div>
<div class="honbun last">
○○○○○○○○○○○○○○○○○○○○○○
</div>
<div id="h002" class="heading">
○○○○○
</div>
<div class="honbun">
○○○○○○○○○○○○○○○○○○○○○○
</div>
<div class="honbun last">
○○○○○○○○○○○○○○○○○○○○○○
</div>
```

適当になんでも同じタグを使って範囲だけ示してしまうと、それらを区別するために余分な属性（赤で示した部分）をつける必要が生じる。また範囲は分かるが、それらが何であるのかは分かりにくい

まず、なんでもかんでも同じタグをつけてしまうと、それらを区別するために後述する「属性」というものを別途追加する必要が出てきます。見出しに対して見出しのタグ、本文の段落に対して段落のタグ、というように当たり前にタグがついていればそのままシンプルに表示指定できるものが、分かりにくくなり逆に複雑化することになるわけです。もちろん、余分なものを追加する分、ファイルの容量も増えます。

▶▶正しいHTMLは万能データ

さらに、ページ内のどの部分にも同じタグがつけられていると、多種多様な機器（今後新しく出現するであろう未知の機器も含む）で利用することが難しくなってしまいます。たとえば、ページの内容を音声で読み上げる場合を考えてみてください。どの部分にも同じタグしかついていなければ、タグは何かしらの範囲を示しているということが分かるだけで、あってもほとんど意味がありません。それでは、タグのない単純なテキストファイルと同じようなものです。

しかし、もし見出しに対して見出しのタグ、本文の段落に対して段落のタグ、というように**適切にタグがつけられていれば、そのタグは大きな意味を持ちます**。たとえば、Webページの内容を音声で読み上げさせる場合には、見出しの前で特定の音を鳴らしてそれが見出しであることを伝えたり、見出しと段落を別の音声で読み上げさせることで区別することも可能になります。また、単純に一定サイズのテキストしか出力できないような機器／ブラウザであっても、見出しは中央揃え、本文は左揃え、といったようにしてそれらを区別できます。つまり、正しく適切にタグが付けられていれば、一般的なブラウザとはまったく異なる環境でも利用できる“**万能データ**”になるということです。そして、それこそがWebページの持つ最大の能力であり特徴でもあるのです。

```
<h1>
○○○○○
</h1>
<p>
○○○○○○○○○○○○○○○○○○○○
</p>
<p>
○○○○○○○○○○○○○○○○○○○○
</p>
<h1>
○○○○○
</h1>
<p>
○○○○○○○○○○○○○○○○○○○○
</p>
<p>
○○○○○○○○○○○○○○○○○○○○
</p>
```


適切にタグがつけられていれば、環境に合わせた表現で出力可能な“万能データ”となる

LECTURE 3-2

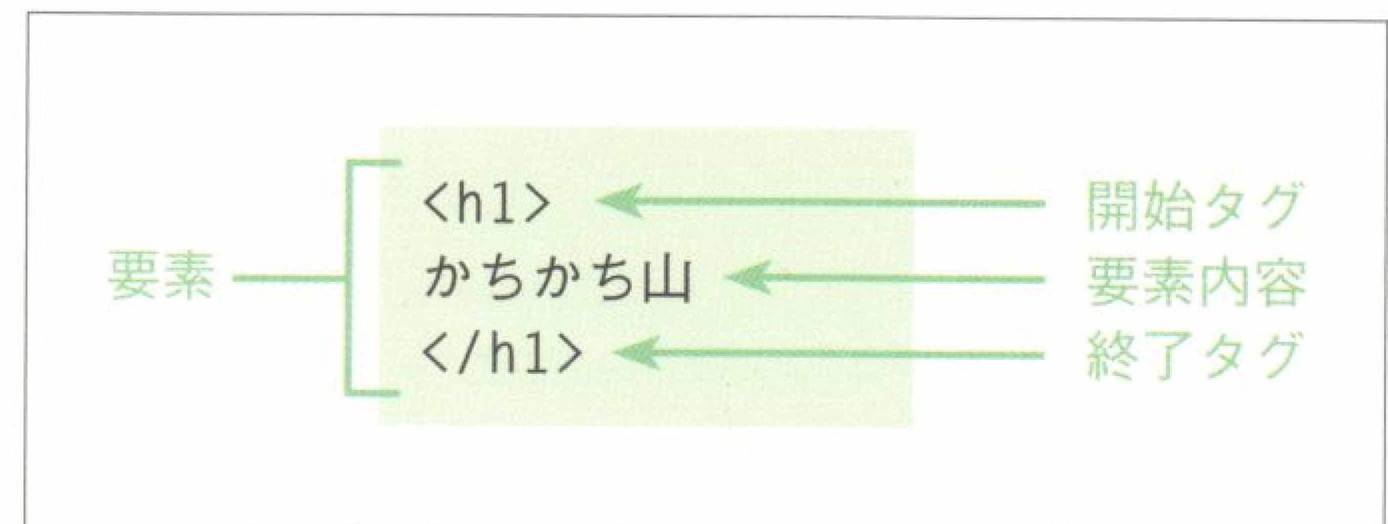
HTMLの基礎知識

HTMLのタグ付けで使ったパーツは、名前や書き方が決まっています。また特殊な書き方をする場合もありますので、覚えておきましょう。

▶▶HTMLの専門用語

これからHTMLの説明をしていくにあたって、その説明が何を指しているのかがハッキリと分かるようにするために、まずはHTMLで使われる用語から覚えていきましょう。

以下は、Chapter 2でテキストにタグをつけたときのソースコードの一部（大見出しとしてタグをつけた部分）です。



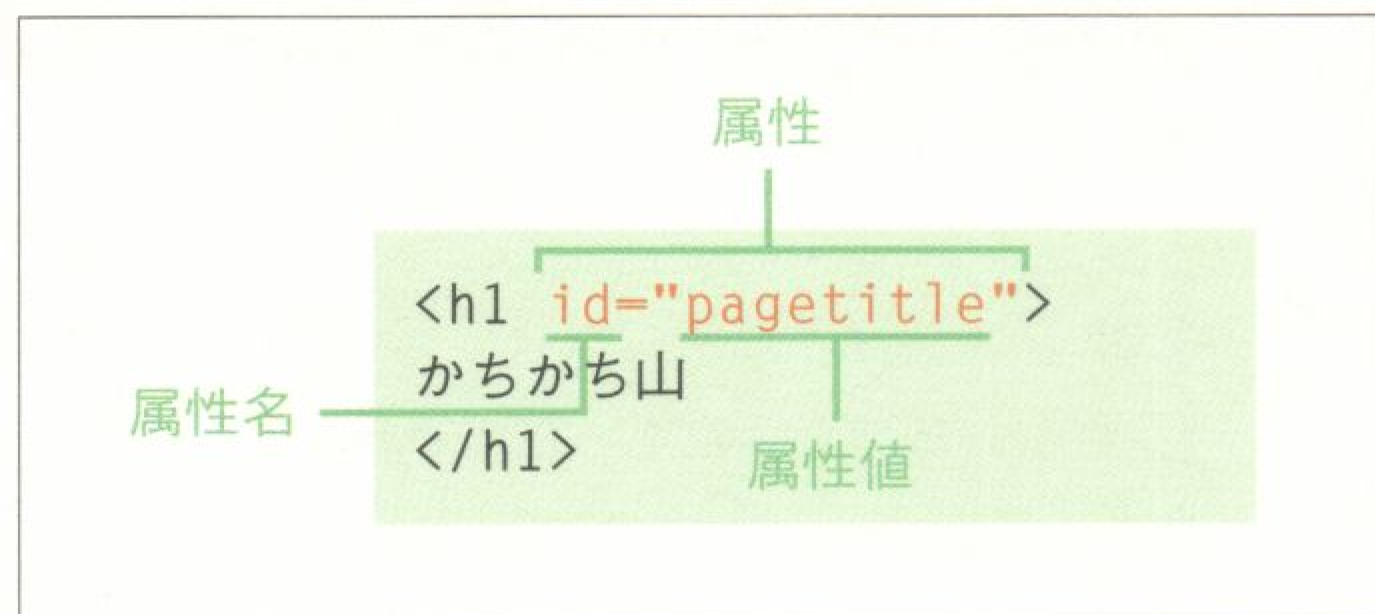
テキストにHTMLのタグをつけたときの各部分の名称

タグはテキストの特定の範囲の前後につけますが、前のタグを**開始タグ**、後ろのタグを**終了タグ**と言います。このとき、開始タグと終了タグの間にはさまれているテキストは**要素内容**、または単に**内容**と呼びます。
そして、開始タグから終了タグまでを含む全体を**要素**と言います（Webページの構成要素という意味です）。この要素という言葉は、「<h1>～</h1>」であればh1要素、「<p>～</p>」であればp要素というように使われます。h1やpの部分だけを指し示す場合は要素名と言います。

▶▶ 属性とは？

要素には、その要素の特性や状態などを示すために**属性**を指定することもできます。属性は「○○○="□□□"」または「○○○='□□□'」の書式で、開始タグの要素名の直後に半角スペースで区切って書き入れます。

属性には、その要素に固有のもの、どの要素にも共通して指定できるものなどがあり、半角スペースで区切って必要な数だけ順不同で指定できます。



属性は「属性名="属性値"」の書式で指定する(「"」の代わりとして「'」も使用可能)。「属性値」は、単に「値」と呼ばれる場合もある

```
<h1 id="pagetitle" class="blog" lang="ja">
かちかち山
</h1>
```

```
<h1 lang="ja" class="blog" id="pagetitle">
かちかち山
</h1>
```

属性は順不同でいくつでも指定できる。
上の2つのソースコードは属性の指定順序が異なるが、意味的にも表示結果にも違いはない

▶▶ 半角スペース・改行・タブの表示

ソースコード上でタグの直前または直後に入れた半角スペース・改行・タブは、ブラウザでの表示に影響を与えません。たとえそれらが複数連続で入っていたとしても、それらが無いかのように表示されます。

たとえば、右のソースコードの表示結果はどれも同じになります(つまり、どの書き方をしてもOKです)。

```
<h1>かちかち山</h1>
```

```
<h1>↵
かちかち山↵
</h1>
```

```
        <h1>↵
        かちかち山↵
        </h1>
```

タグの直前または直後に半角スペース・改行・タブを入れても、表示には影響しない

また、開始タグと終了タグの間、つまり要素の内容として入れたテキスト内部の改行やタブは、ブラウザで表示されるときには1つの半角スペースに置き換わります。
要素内容として入れたテキストの途中で改行させたい場合には、Chapter 5で登場する改行用のタグが別途必要となる点に注意してください。

また、半角スペース・改行・タブのうちいずれかを連続して入れた場合は、それらはまとめて半角スペース1つになって表示されます(これは単語を半角スペースで区切って表示する英語のような言語のテキストを適切に表示するためです)。

```
<h1>Win Win Yama</h1>

        <h1>↵
        Win↵
        Win↵
        Yama↵
        </h1>
```

はじめのh1要素の内容は単純に半角スペース1つで区切ってあり、2つ目のh1要素は単語間に改行と8つの半角スペースを入れてある



上のソースコードをHTMLの枠組みの中に入れてブラウザで表示させたところ。改行とそれに続く8つの半角スペースは、1つの半角スペースに置き換えられている

▶▶ 特別な書き方が必要となる文字

HTMLのソースコードの中に半角の「<」または「>」を記入すると、タグの一部だと認識されます。そのため、文字として「<」または「>」を表示させたい場合には、表のような特別な書き方をする必要があります。なお、この書き方をする場合は、同じアルファベットでも小文字と大文字は別の文字として扱われますので、必ずこの表の通りに小文字で書いてください。

ソースコード上の書き方	ブラウザでの表示
<	<
>	>

半角の「<」または「>」を表示させたい場合には、このように書く

このように、HTMLにおいて特別な役割を持っていたり、キーボードから直接入力できないような特殊な文字を示すには、&○○○;という書式を使います。ちなみに、「<」を示すために使われて

いる「lt」は「less than（より小さい）」の略で、同様に「gt」は「greater than（より大きい）」の略です。

&○○○；という書式が特別な意味を持っているため、「<」「>」と同様に半角の「&」を表示させたい場合にも特別な書き方をする必要があります。また、属性は「属性名="属性値"」の書式で書き込みますが、属性値の中に「"」を書き込むとそこで属性値が終了したと見なされてしまいます。そこで、属性値の中にも「"」を入れられるように、これにも特別な書き方が用意されています。

ソースコード上の書き方	ブラウザでの表示
`&`	&
`"`	"

半角の「&」を表示させたい場合や、半角の「"」を属性値の中で使用したい場合には、このように書く

ここで紹介した &○○○; という書式で入力できる文字は、このほかにもたくさんあります。入力可能なすべての文字の一覧は仕様書にありますので、必要になった場合には以下のページで確認してください。

8.5 Named character refere… ×
www.w3.org/TR/html5/named-character-references.html
W3C Working Draft

8.5 Named character references

This table lists the character reference names that are supported by HTML, and the code points to which they refer. It is referenced by the previous sections.

Name	Character(s)	Glyph
AElig;	U+000C6	Æ
AMP;	U+00026	&
Aacute;	U+000C1	Á
Abreve;	U+00102	Ă
Acirc;	U+000C2	Â
Acy;	U+00410	А
Afr;	U+1D504	⊠
Agrave;	U+000C0	À
Alpha;	U+00391	Α
Amacr;	U+00100	Ā
And;	U+02A53	⩓
Aogon;	U+00104	Ą
Aopf;	U+1D538	⊠
ApplyFunction;	U+02061	
Aring;	U+000C5	Å
Ascr;	U+1D49C	⊠
Assign;	U+02254	≔
Atilde;	U+000C3	Ã
Auml;	U+000C4	Ä
Backslash;	U+02216	∖
Barv;	U+02AE7	⫧
Barwed;	U+02306	⌆
Bcy;	U+00411	Б

hyphen;	U+02010	‐
iacute;	U+000ED	í
ic;	U+02063	
icirc;	U+000EE	î
icy;	U+00438	и
iecy;	U+00435	е
iexcl;	U+000A1	¡
iff;	U+021D4	⇔
ifr;	U+1D526	⊠
igrave;	U+000EC	ì
ii;	U+02148	ⅈ
iiiint;	U+02A0C	⨌
iiint;	U+0222D	∭
iinfin;	U+029DC	⧜
iiota;	U+02129	℩
ijlig;	U+00133	ĳ
imacr;	U+0012B	ī
image;	U+02111	
imagline;	U+02110	
imagpart;	U+02111	
imath;	U+00131	ı
imof;	U+022B7	⊷
imped;	U+001B5	Ƶ
in;	U+02208	∈

ほかに使用できる書き方はHTML5の仕様書に掲載されている
（http://www.w3.org/TR/html5/named-character-references.html）

▶▶ コメントの書き方

HTMLのソースコードの中には、表示に影響することのないコメント(注釈やメモのようなもの)を書き込んでおくことができます。HTMLでは、下のように <!-- と --> で囲った範囲がコメントとなります。


```
<!--コメント-->
```

```
<!--
コメント
-->
```

コメントの書き方。<!-- と --> で囲われた範囲がコメントとなる

たとえば、ソースコード上で開始タグと終了タグが極端に離れているような場合に、終了タグの直後にコメントを入れて対応する開始タグがどれなのかが分かるようにしておく、といった使い方をします。

<!-- と --> の中には自由にテキストを書き込んでおくことができますが、複数のハイフン(-)を連続して入れると、そこでコメントの内容が終了していると見なされますので注意してください(コメントの途中に -- が入っていると、ブラウザによっては表示に影響が出る場合もあります)。

```
<!------------コメント------------->
```

コメントの書き方の悪い例。連続した複数のハイフンを入れてはいけない

LECTURE 3-3

HTMLのバージョンについて

HTMLのバージョンについて説明します。本書で解説していく「HTML5」がどういう経緯で登場してきたのか、以前のバージョンとどう違うのかを簡潔にまとめてみました。

▶▶HTML4.01とXHTML1.0

ところで、HTMLにはいくつかのバージョンと種類があって、それぞれで使用できる要素や属性、細かい部分での書き方のルールなどが異なります。現在、一般的に利用されているHTMLには、次のようなものがあります。

バージョン	種類	説明
HTML4.01（1999年に完成・公開）	Strict	正規のHTML4.01
	Transitional	文法的にあまいHTML4.01
	Frameset	フレーム機能が使えるHTML4.01
XHTML1.0（2000年に完成・公開）	Strict	正規のXHTML1.0
	Transitional	文法的にあまいXHTML1.0
	Frameset	フレーム機能が使えるXHTML1.0
HTML5（2012年現在は草案）	なし	現在策定中のHTMLの最新版

一般的に利用されているHTMLの種類

HTML4.01は、現時点で仕様が確定しているHTMLの最新版です。これはSGMLという国際標準規格によって作られたものですが、それをWeb向けの新しい規格であるXMLで定義し直したものがXHTML1.0です。したがって、細かい部分での書き方が違うだけで、使える要素と属性はHTML4.01と同じです。現時点では、おそらくXHTML1.0によって作られているWebページがもっとも多いのではないかと思われます。

HTML4.01とXHTML1.0は、さらに「Strict」「Transitional」「Frameset」という3種類に分かれています。HTML4.01とXHTML1.0の本来の仕様は「Strict」です。しかし、HTML4.01の仕様が完成した1999年頃は、まだCSSが普及しておらず、文法などはほとんど無視したHTMLで表示指定を行うのが一般的でした。しかも、CSSに十分に対応したブラウザも、当時はまだ存在していませんでした。そのような状況において、HTML4.01の仕様通りのWebページを制作することは現実的に困難であることが予想されたため、本来なら正しくない記述も多少は認めた暫定仕様で

ある「Transitional」も用意されたのです。そして、その暫定仕様に加えて、ウィンドウを分割して表示できるフレーム機能を追加した仕様が「Frameset」です。

HTML5

現在では多くのブラウザがCSSに十分対応し、CSSで表示指定をするのが一般的となっているため、HTML5からは「Transitional」はなくなっています。また、フレームはほとんど利用されなくなっていますので、HTML5には「Frameset」も用意されていません(ウィンドウを分割するのではなく、ウィンドウの中に別のページを組み込むインラインフレームはHTML5でも利用できます)。本書では、この最新の仕様であるHTML5に準拠してHTMLを解説していきます。

ただし、HTML5はまだ策定途中の仕様であるため、現在でも微調整が続けられて仕様が変更されています。そのような状況であるため、ブラウザのサポートも完全ではありません。とはいえ、広く一般的に利用されている部分については、すでに利用可能となっていますので、本書ではそのような主要な部分を中心に解説していきます。

COLUMN

大文字と小文字の区別

HTML5もCSS3も、基本的には大文字と小文字を区別しません(id属性の値のような一部の例外を除く)。したがって、要素名や属性名などは大文字で書いても小文字で書いてもどちらでもかまいません。しかし、文法的に間違いではないとはいえ、統一性のないソースコードは読みにくいものです。また、HTML5はまだ最終的に確定した仕様ではなく、将来的に大文字と小文字を区別するようになる可能性もゼロとは言えません。世界中の多くのページが現在そうなっているように、通常は小文字で統一して書いておくのが良いでしょう。

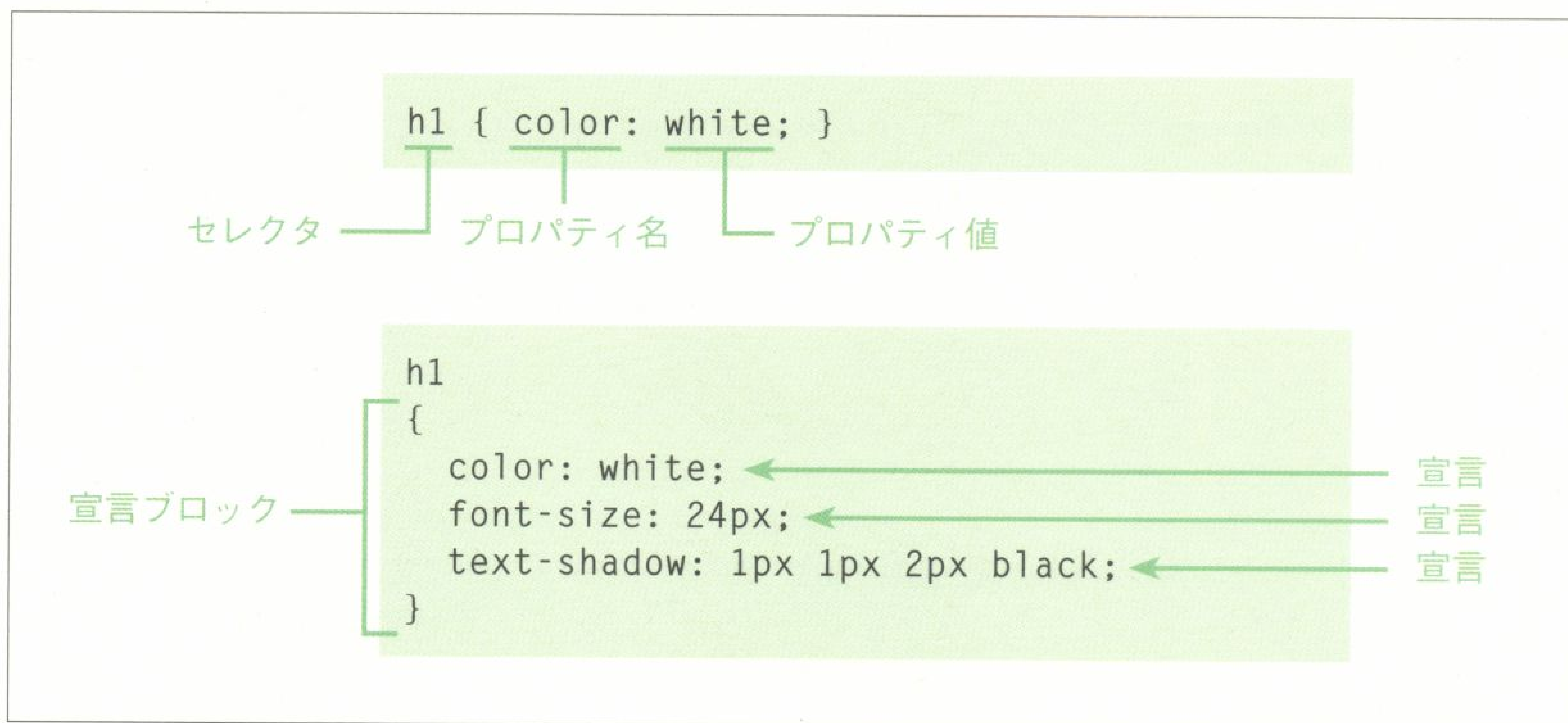
LECTURE 3-4

CSSの基礎知識

続いてCSSについて説明します。CSSの書式を構成する各部分にも名前がついていますので覚えておきましょう。書き方のルールについても紹介します。

▶▶CSSの専門用語

次はCSSです。まずはCSSで使われる専門用語から説明します。



CSSのソースコードの各部分の名称

CSSは、表示指定をどの要素に対して適用するのかを最初に示しますが、その部分のことを**セレクタ**(selector)といいます。
そして、その要素に対する具体的な表示指定は、セレクタに続く { } の中に書いていきます。1つひとつの表示指定(「何をどのようにするか」を示す部分)は**宣言**と言い、{ } を含むその範囲全体を**宣言ブロック**と言います。
1つの宣言は、**プロパティ名**(「何を」を示す部分)と**プロパティ値**(「どのようにする」を示す部分)を順にコロン(:)で区切って指定します。通常はその直後にセミコロン(;)をつけますが、これは宣言と宣言を区切るための記号なので、宣言ブロック内の最後の宣言にはつけなくてもかまいません(ソースコードのコピー&ペーストをしたときにセミコロンを付け忘れるミスを防ぐ目的で、一般的には最後の宣言にもセミコロンをつけておきます)。

▶▶ 書き方のルール

セレクタ、プロパティ名、プロパティ値、{ } : ; の各記号の前後には、半角スペース・改行・タブを自由に入れることができます。したがって、次のようなさまざまな書き方が可能です。

```
h1 {
  color: white;
  font-size: 24px;
}
```

```
h1
{
          color       :  white   ;
          font-size   :  24px    ;
          }
```

```
h1{color:white;font-size:24px;}
```

書式を構成する各部分の間には、半角スペース・改行・タブを入れることができる

どのような書き方をするのも自由ですが、いつも同じ書き方をしておかないとコピー＆ペーストによって書式がバラバラになったり、予想外の編集ミスを招いたりすることがあります。いつも一定のパターンで統一させて書くことが効率アップにつながるということを覚えておいてください。

また、CSSの適用対象を示すセレクタは、カンマ(,)で区切って同時に複数を指定できます。たとえば、h1要素、h2要素、p要素に同時に同じ表示指定をする場合は、次のように書きます。

```
h1, h2, p {
  color: white;
  font-size: 24px;
}
```

```
h1,
h2,
p
{
  color: white;
  font-size: 24px;
}
```

複数の適用先に同時に同じ表示指定をするには、セレクタをカンマで区切って指定する

▶▶ コメントの書き方

HTMLと同様に、CSSのソースコード内にもコメントを書き込むことができます。CSSでは /* と */ で囲った範囲がコメントとなります。
ソースコード内に説明や注釈を入れておきたい場合や、表示指定を一時的に無効にしたいときなどに利用できます。

```
/*コメント*/
```

```
/*
コメント
*/
```

CSSのコメントの書き方。/* と */ で囲われた範囲がコメント

LECTURE 3-5

CSSのバージョンについて

CSSにもバージョンがあります。ただし、HTMLのバージョンとは少し意味合いが異なりますので、注意してください。

▶▶ 現在使用されているのはCSS2.1とCSS3の一部

HTMLと同様に、CSSにもいくつかのバージョンがあります。正確に言うと、CSSの場合は“バージョン”ではなく“レベル”という用語が使われており、新しいレベルのCSSは古いレベルのCSSを元にして機能の追加と改善を行うことになっています。つまり、「レベルが変わると仕様が変わる」ということではなく、基本的には「新しいレベルはこれまでの仕様をそのままにして機能を追加したもの」であるということです。ただし、そのような意味を除けば、レベルという用語も基本的にはバージョンのような意味あいで使われています。

CSSには、もっとも古いレベルであるCSS1、その次のレベルであるCSS2、それを現実の状況に合わせて再調整したCSS2.1、そして、それを元に仕様書を機能別に分割して大幅な機能追加を行っているCSS3という4つのレベルがあります。CSS3になって仕様書を分割したのは、それまでのように1つの仕様書にしていると、内容が膨大すぎてなかなか完成させることができなくなったからです。機能別にモジュール化しておけば、状況の変化に応じてより早く仕様書を改訂することが可能になります。

現時点での基本となるCSSは、CSS2.1です。CSS1とCSS2は、現実の状況に合わせて調整されたCSS2.1によって置き換えられたと考えておけばよいでしょう。CSS3はまだ一部の機能(モジュール)の仕様しか確定していませんが、それらを含む多くの機能がすでに使用されはじめています。つまり、現在使用されているCSSは、CSS2.1とCSS3の一部であるということになります。本書では、CSS3については、現時点で使用可能な(比較的多くのブラウザにサポートされている)ものを取り上げ、それ以外の部分ではCSS2.1を使って解説していきます。

▶▶ すべてのブラウザで同じに見えなくても良い

ただし、IE6のようにCSS3の登場する以前にリリースされたブラウザは、当然ながらCSS3には対応していません。CSS3の一部の機能については、そのようなブラウザの独自拡張機能やスクリプトなどで同様に表示させることも可能ですが、それによって弊害が起きることもあります。CSS3を導入する際の基本は、まずCSS2.1できちんと見られるように作っておき、CSS3に対応したブラウザで見るとさらに良く見えるようにする、という考え方で制作することです(これを「プログレッシブ・エンハンスメント」と言います)。

かつてはどのブラウザでも同じように見えるようにすることが当然だと思われていた時期もありましたが、今は新しいブラウザで見るとカドが丸くなっているけれども、古いブラウザで見るとカドは丸くなっていない、というようにまったく同じにすることにはこだわらずに作るのが主流です(特に海外の大手サイトはそのようになっています)。現在では、CSS3の機能を使って、異なる環境や画面サイズの場合はあえて異なる表示指定を適用して違う表示にすることも多くなってきています。少なくとも、どのブラウザで見てもまったく同じようにする必要はなく、古いブラウザで見ても内容がきちんと分かるようになっていればOK、というのが現代の考え方だということを覚えておいてください。

CHAPTER 4

ページ全体の枠組み

Chapter 4ではHTMLのページ全体の枠組みを構成する各要素と、そこにCSSを組み込む方法を解説します。そして、背景関連のプロパティを使って、ページ全体の背景を指定するところまでを学習します。これ以降はHTMLの要素とCSSのプロパティがどんどん登場してきますので、1つひとつしっかりと覚えていきましょう。

LECTURE 4-1

HTMLの全体構造

Chapter 2では、テキストにタグをつけたあとに、それらをページ全体の枠組みとなるタグの中に入れました。ここではまず、その枠組みの各部分について説明します。ここに書かれているタグは、すべてが文法的に必須というわけではありませんが、一般的なHTMLであれば通常はすべてが使用されます。
それでは、この枠組みの各部分を1つずつ確認していきましょう。

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>サンプル</title>
</head>
<body>

</body>
</html>
```

Chapter 2で使用したHTMLのページ全体の枠組みとなるタグ

▶▶ <!DOCTYPE html> 【HTML5新】

まずは先頭にある、普通のタグとはちょっと違った雰囲気のタグからです。実はこれはHTMLの中では特別な役割を持ったもので、要素のタグではなくDOCTYPE宣言(文書型宣言)と呼ばれているものです。

HTML5より前の各バージョンのHTMLには、それぞれにDTD(Document Type Definition:文書型定義)と呼ばれるファイルが用意されていました(Webの仕様を作成しているW3Cという組織のサイト内にあります)。DTDには、そのバージョンのHTMLではどの要素・属性が使用できるのか、各要素はどこに何回どの順序で配置できるのか、といった情報が独自の文法で記述されています。そして、そのDTDをHTMLの先頭にあるDOCTYPE宣言で指定することで、そのHTMLファイルがどのバージョンの文法にしたがって記述されているのかを示すルールになっていたのです(そのような意味で文書型宣言という名前になっています)。

しかし、2000年頃から、各社ブラウザはこのDOCTYPE宣言を別の用途にも利用するようになっていきました。当時、HTMLで表示指定をしていたページのほとんどは文法を無視しており、DOCTYPE宣言はつけていなかったので、DOCTYPE宣言がなければ「HTMLで表示指定をしている旧式のページ」、DOCTYPE宣言があれば「標準仕様に準拠した新しいページ」という風に判定し、ブラウザの表示モードを自動的に切り替えるようになったのです。はじめにMac版の

Internet Explorerがその方式を採用すると他社のブラウザもそれにならい、現在ではほとんどのブラウザが同様の仕組みになっています。

しかし、HTML5からはDTDはなくなりました。もともとDTDに書かれていた情報は、HTML5の仕様書の中に専用の言語を使用せずに書き込まれることになったからです。そのため、本来の意味からすればHTML5にはDOCTYPE宣言は不要となったのです。ところが現在のほとんどのブラウザは、DOCTYPE宣言をつけなければ「HTMLで表示指定をしている旧式のページ」と認識してそのページを扱ってしまうことになります(その場合、古いブラウザの独自仕様などに合わせた表示となるので正しい表示結果は得られなくなります)。そうなることを回避する目的で(つまりブラウザの表示モードを制御するだけの目的で)、HTML5では必要最低限のシンプルなDOCTYPE宣言をつけることとなっています。

なお、本書では `<!DOCTYPE html>` のように一部を大文字で記述していますが、DOCTYPE宣言は大文字で書いても小文字で書いてもOKです。

---

html要素

DOCTYPE宣言のあとには `<html>` ～ `</html>` を配置します。それ以外の要素はすべてその中に書き込みます。

ある要素の中に直接入っている要素をその要素の**子要素**であると表現することがあります。逆に、自分を直接含んでいる要素は**親要素**ということになります。HTMLの各種要素をこのような親子関係で考えたとき、html要素はすべての要素の先祖ということになりますので**ルート要素**(the root element)とも呼ばれています。

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>サンプル</title>
</head>
<body>

</body>
</html>
```

DOCTYPE宣言を除くすべての要素はhtml要素の中に入れる

COLUMN

HTML4.01とXHTML1.0のDOCTYPE宣言

参考までに、HTML4.01とXHTML1.0のDOCTYPE宣言を掲載しておきます。HTML5より前のバージョンでは、このような長いDOCTYPE宣言が使用されていました。バージョンが同じでも、種類(Strict / Transitional / Frameset)によってDOCTYPE宣言が異なっている点に注意してください。

HTML4.01 Strict の場合

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
```

HTML4.01 Transitional の場合

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
 "http://www.w3.org/TR/html4/loose.dtd">
```

HTML4.01 Frameset の場合

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
 "http://www.w3.org/TR/html4/frameset.dtd">
```

HTML4.01のDOCTYPE宣言

XHTML1.0 Strict の場合

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

XHTML1.0 Transitional の場合

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

XHTML1.0 Frameset の場合

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
```

XHTML1.0のDOCTYPE宣言

▶▶ head要素とbody要素

html要素以外の要素はすべてhtml要素の中に書き込みますが、html要素の中に直接入れることができるのは、head要素とbody要素だけです。かならずhead要素・body要素の順で1つずつ入れる決まりになっています。

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>サンプル</title>
</head>
<body>

</body>
</html>
```

html要素の中には、head要素とbody要素を順に1つずつ入れる

head要素は、そのWebページに関する情報を入れるための要素です。
たとえば、文字コードを示す要素、Webページのタイトルを示す要素、適用するCSSファイルのURLを示す要素、適用するJavaScriptファイルのURLを示す要素、といった要素を順不同で必要なだけ入れることができます。head要素に入れた内容は、基本的にはWebページの内容としてブラウザで表示されることはありません(ただし、タイトルはウィンドウのタイトルバーやタブなどに表示されます)。

それに対してbody要素には、ブラウザで表示させたい内容を入れます。つまり、ブラウザでWebページのコンテンツとして表示されるのは、body要素の中に入れられた内容なのです。body要素にも、必要な要素を順不同でいくつでも入れることができます。

このような、どの要素の中にどの要素をどの順番でいくつ入れられるか、というルールは要素ごとにあらかじめ決められています。本書では、巻末のAppendixで一覧表で掲載していますので、必要に応じて参照してください。

▶▶ title要素

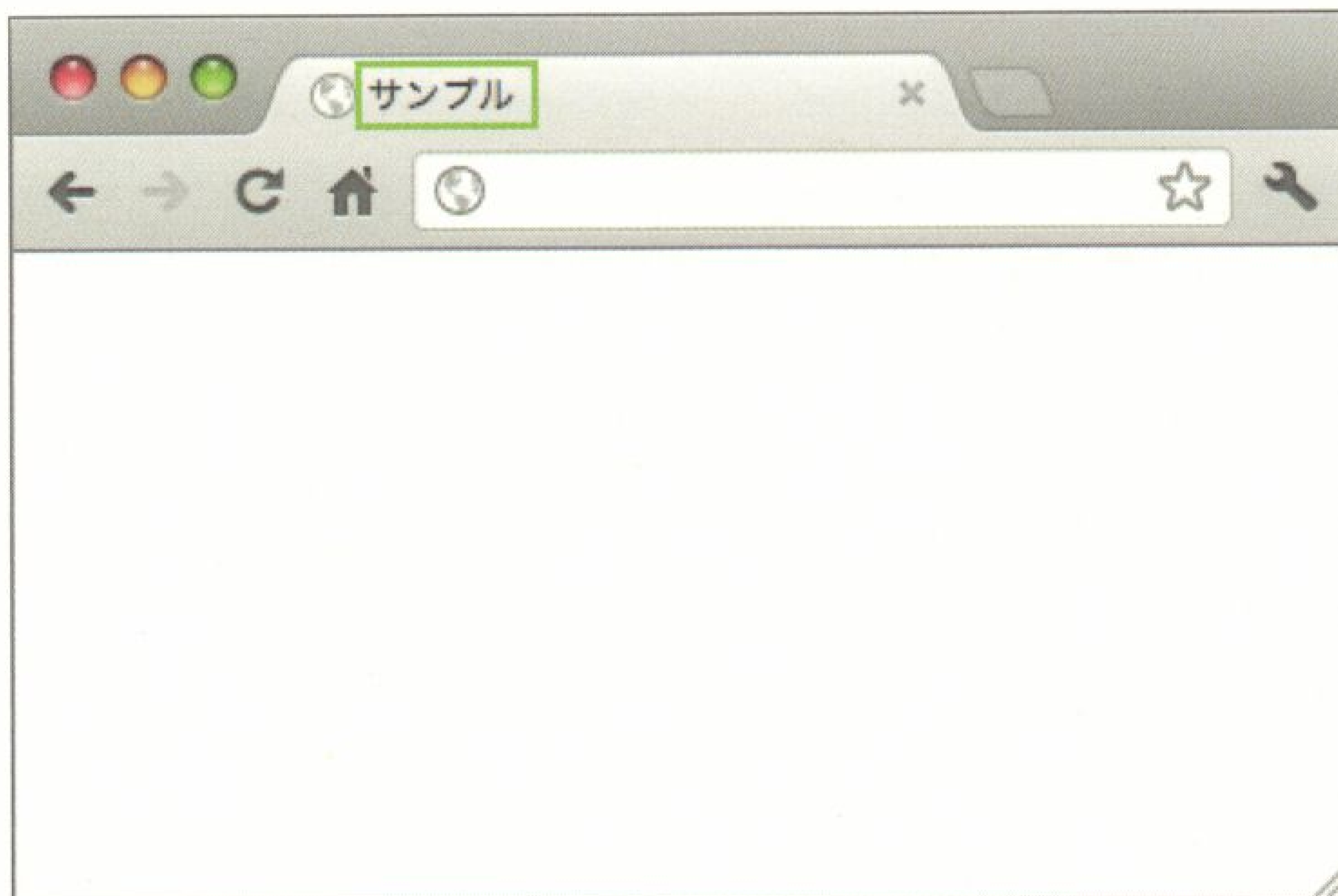
title要素は、そのWebページのタイトルであるテキストを入れる要素です。テキスト以外の他の要素は入れられませんので注意してください。

title要素の内容は、Webページを表示する際にブラウザのウィンドウのタイトルバーまたはタブに表示されます。この要素は、head要素の中に必ず1つだけ入れる必要があります(head要素内であれば入れる場所は自由です)。

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>サンプル</title>
</head>
<body>

</body>
</html>
```

Webページのタイトルはtitle要素の内容として書き込む



title要素の内容は、このようにタブやタイトルバーに表示される

▶▶ meta要素 【HTML5改】

meta要素は、そのWebページ自身に関するさまざまな情報（メタデータ）を示すことのできる要素です。
ここまでに紹介してきた要素とは異なり、meta要素は各種情報を要素内容ではなく属性の値に書き込んで示します。そのため、meta要素には常に要素内容がなく、終了タグもありません。HTMLにはこのようなタイプの要素がいくつかあり、それらは**空要素**と呼ばれています。

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">   ← meta要素は空要素なので開始タグのみ
<title>サンプル</title>
</head>
<body>

</body>
</html>
```

meta要素には、要素内容と終了タグがない

空要素は開始タグだけを書けばいいのですが、XHTMLの書式との互換性を持たせるために、右のように開始タグの最後の > の直前に（空白スペース）/ を入れることもできます。

```
<meta charset="utf-8">

<meta charset="utf-8" />
```

スペース＋/を入れてもOK

HTML5の空要素は、このような2種類の書き方ができる

meta要素には、次の属性を指定することができます。
「文字コード」を指定する場合は、charset属性を使用します。「文字コード」以外の情報を指定する場合は、name属性またはhttp-equiv属性のいずれかで「情報の種類」を示し、具体的な情報はcontent属性の値として指定します。name属性とhttp-equiv属性のどちらを使用するかは、情報の種類によって異なります。

meta要素に指定できる属性

- **charset="文字コード"**
  このHTMLファイルがどの文字コードで保存されているのかを示します。
  UTF-8の場合は「UTF-8」、シフトJISの場合は「Shift_JIS」、EUCの場合は「EUC-JP」のように指定します（HTML5ではUTF-8で保存することを推奨しています）。
  文字コードを指定する際は、大文字と小文字を区別せずに使用できます。

meta要素に指定できる属性（続き）

- **name="情報の種類"**
  content属性で指定する情報の種類を名前で示します。
  情報の種類によってはhttp-equiv属性を使用します。

- **http-equiv="情報の種類"**
  content属性で指定する情報の種類を名前で示します。
  情報の種類によってはname属性を使用します。

- **content="情報"**
  name属性またはhttp-equiv属性で指定された種類の具体的な情報を指定します。

meta要素の典型的な使い方を示しますので参考にしてください。

```
<head>
・・・
<meta name="generator" content="WordPress 3.4.1">  …Ⓐ
<meta name="description" content="札幌のおいしいお店を紹介するブログです。">  …Ⓑ
<meta http-equiv="content-type" content="text/html; charset=Shift_JIS">  …Ⓒ
・・・
</head>
```

meta要素の典型的な使用例

meta要素は、head要素内の任意の位置で何度でも使用できます。

Ⓐは、そのWebページを生成したソフトウェアが「WordPress 3.4.1」であることを示しています。

ⒷはWebページの簡単な紹介文です。この文章は、検索結果としてこのページが一覧表示されるときに利用される場合があります。

Ⓒは文字コードを指定する古い方法です。charset属性に未対応のブラウザ向けに指定できますが、この方法で文字コードを指定した場合は、charset属性による文字コード指定は入れられません。

LECTURE 4-2

CSSの組み込み方

枠組みの各部分の意味と役割が分かったところで、次はそのHTMLにCSSを組み込むための3種類の方法を説明します。1つめはChapter 2でCSSファイルを読み込ませたときのlink要素を使う方法で、2つめはstyle要素を使ってHTMLファイルの中に直接CSSを書き込む方法、3つめはstyle属性を使用して属性値としてCSSを書き込む方法です。

▶▶link要素

Chapter 2では、link要素を次のように使用してCSSファイルを読み込ませました(link要素は空要素です)。このように、rel属性の値にはキーワード「stylesheet」を指定し、href属性の値としてCSSファイルのURLを指定することで、CSSファイルを読み込ませることができます。link要素は、meta要素と同様にhead要素内の任意の位置で何度でも使用できます。つまり、link要素を複数使用することで、複数のCSSファイルを読み込ませることもできるということです。

sample/chapter-04/lecture-4-2/01.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>サンプル</title>
<link rel="stylesheet" href="style.css">
</head>
<body>

</body>
</html>
```

Chapter 2でCSSファイルを読み込ませたときの指定

実際には、link要素はCSSファイルを読み込ませるだけでなく、そのWebページに関連するさまざまなファイルを示すことのできる要素です。rel属性にはそのファイルの種類を表すキーワードを指定し、そのファイルのURLをhref属性で示します。しかし、現実的にはCSSファイルを読み込ませる以外の用途ではあまり使用されていません(ほかの用途で使用しても未対応で機能しないブラウザが多いためです)。link要素に指定できる属性は次ページの通りです。

link要素に指定できる属性

・rel="ファイルの種類"　※必須

関連するファイルの種類をキーワードで示します。
指定できる値は次の通りです(主なもののみ抜粋)。なお、この値は大文字で書いても小文字で書いてもかまいません。

値	意味
stylesheet	スタイルシート(CSS)
alternate	代替バージョン(異なる言語や異なる媒体向けなど)
prev	連続しているページ中の前のページ
next	連続しているページ中の次のページ

・href="ファイルのURL"　※必須

関連するファイルのURLを指定します。

・media="適用対象"

CSSを適用する対象の出力媒体(パソコン画面・プリンタ・テレビなど)を限定したい場合に指定します。
下の表に掲載されている値が指定できますが、さらに細かく複雑な指定方法もあります。詳細は Chapter10の「メディアクエリー」で解説します。この属性を指定しなかった場合は、「all」が指定された状態となります。なお、この値は大文字で書いても小文字で書いてもかまいません。

値	意味
all	すべての機器
screen	パソコン画面
print	プリンタ
projection	プロジェクタ
tv	テレビ
handheld	携帯用機器(画面が小さく回線容量も小さい機器)
tty	文字幅が固定の端末(テレタイプやターミナルなど)
speech	スピーチ・シンセサイザー(音声読み上げソフトなど)
braille	点字ディスプレイ
embossed	点字プリンタ

・type="MIMEタイプ"

関連するファイルのMIMEタイプを指定できます。この属性を指定しなかった場合は「text/css」が指定された状態となります。

COLUMN

CSSファイルの文字コードの指定方法

HTMLもCSSもUTF-8で保存されているのであれば特に必要はありませんが、HTMLとCSSの文字コードが違う場合にはCSS側にも文字コードを指定した方が安全です。CSSで文字コードを指定するには、ソースコードの先頭に次のように記述します（必ずファイルの先頭に記述する必要があります）。

```
@charset "文字コード";
```

文字コードとして指定できるのは、meta要素のcharset属性の値と同様です。
UTF-8の場合は「UTF-8」、シフトJISの場合は「Shift_JIS」、EUCの場合は「EUC-JP」のように指定します。

▶▶ style要素

link要素は、CSSが書かれた別のファイルを読み込むために使用しましたが、head要素の中にstyle要素を配置すると、要素内容として直接CSSを記入することができます。

sample/chapter-04/lecture-4-2/02.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>サンプル</title>
<style>
body { background: orange; }
h1, p { color: white; }
p { font-size: 18px; }
</style>
</head>
<body>
<h1>かちかち山</h1>
<p>
昔々、ある会社に山田という名前の若い社長さんがおりました。社長仲間のあいだでは「やま」と呼ばれていました。山田社長には、ことあるごとに「Win-Winの考え方が大切なんです」と語りだすクセがありました。
</p>
</body>
</html>
```

style要素を使用すると、内容としてCSSを直接記入できる

しかし、style要素を使用すると、HTMLファイルの中に表示の指定を組み込んでしまうことになり、他のHTMLとCSSの表示指定を共有できなくなってしまいます。そのため、この方法はこの要素を使用すべきなんらかの理由がある場合（ブログサービスでHTMLのテンプレートしか変更できない場合など）を除いて、それほど多くは使用されていません。link要素と同じ次の属性が指定できます。

style要素に指定できる属性

- **media="適用対象"**
  CSSを適用する対象の出力媒体（パソコン画面・プリンタ・テレビなど）を限定したい場合に指定します。指定できる値はlink要素のmedia属性と同様です。この属性を指定しなかった場合は、「all」が指定された状態となります。なお、この値は大文字で書いても小文字で書いてもかまいません。

- **type="MIMEタイプ"**
  スタイルシート言語のMIMEタイプを指定できます。style要素はもともとCSS専用ではなく、CSS以外のスタイルシート言語にも対応できるようになっているため、この属性が用意されています。この属性を指定しなかった場合は「text/css」が指定された状態となります。

▶▶ style属性

任意の要素にstyle属性を指定して、その値としてCSSの宣言（プロパティ： 値;）部分を書き込むことができます。その際、指定した宣言はstyle属性を指定した要素に適用されますので、セレクタおよび宣言ブロックの範囲を示す記号 { } は不要です。

sample/chapter-04/lecture-4-2/03.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>サンプル</title>
</head>
<body style="background: orange;">
<h1 style="color: white;">かちかち山</h1>
<p style="color: white; font-size: 18px;">
昔々、ある会社に山田という名前の若い社長さんがおりました。社長仲間のあいだでは「やま」と呼ばれていました。山田社長には、ことあるごとに「Win-Winの考え方が大切なんです」と語りだすクセがありました。
</p>
</body>
</html>
```

style属性を使用すると、属性値としてCSSが直接記入できる

ただし、この方法を使用すると、HTMLのあちらこちらに細かく表示指定を埋め込んでしまうことになりメンテナンス性も低下しますので、通常は使用されません。主にちょっとした実験やテストをする際などに使用するものだと考えた方がいいでしょう。

COLUMN

CSSの中にさらに別のCSSを読み込む

HTMLの中からCSSファイルを読み込むにはlink要素を使用しましたが、同様にCSSの中からさらに別のCSSファイルを読み込むこともできます。その際に使用するのが @import で、読み込むCSSファイルのURLは次の例のように "○○○" または url("○○○") の書式で指定します。

```
@import "style.css";
@import url("style.css");
```

URL部分は上のどちらの書式で書いてもよい

また、この書式には適用対象とする出力媒体(link要素やstyle要素に指定できるmedia属性の値と同じもの)を指定することもできます。複数の出力媒体を適用対象とする場合は、それらをカンマ(,)で区切って指定します。

```
@import url("print.css") print;
@import url("tv.css") tv, projection;
```

HTMLのmedia属性の値と同じものを指定することも可能

この @import は、link要素で読み込むCSSファイルの中ででも使用できますし、style要素の内容として書き込むCSSの中ででも使用できます。ただし、必ずCSSによる表示指定よりも前で指定する必要がある点に注意してください。@charsetがある場合は、@charsetを先頭にして、その直後に指定します。

LECTURE 4-3

グローバル属性

HTML5の属性の中には、style属性のようにどの要素にも共通して指定できる属性があります。そのような属性は**グローバル属性**と言い、HTML5では表の15種類が定義されています※5。

属性名	値
id	固有の名前
class	種類を示す名前
title	補足情報
lang	言語コード
style	CSSの宣言
dir	文字表記の方向
accesskey	ショートカットキー
tabindex	タブキーによる移動の順序
hidden	非表示
spellcheck	スペルチェックの有効・無効
contextmenu	コンテキストメニューのID
draggable	ドラッグの可・不可
dropzone	ドロップ時の挙動（copy/move/link）
contenteditable	編集の可・不可
translate	翻訳するかどうか（yes/no）

グローバル属性の中には、定義はされているもののまだ実際の制作ではほとんど使用されていないものもありますので、ここでは一般的に使用されている主要なものをピックアップして紹介しておきます。

※5：実際には、このほかにJavaScriptで使用できる50種類以上のイベントハンドラも定義されています。

一般的に使用されているグローバル属性

▶▶ id属性

要素に対して固有の名前をつける場合に使用します。id属性によってつけられた名前は、CSSでその要素だけに表示指定をする場合や、リンクによってページ内のその位置までジャンプさせる場合などに使用されます。id属性はページ内の特定の1つの要素(またはその場所)を示すものですので、同じページ内の別の場所で同じid属性の値を指定することはできません。id属性の値は、同じアルファベットでも大文字と小文字は別の文字として扱われますので注意してください。

▶▶ class属性

要素の種類を示す(分類としての名前をつける)場合に使用します。あくまで種類を示すための属性ですので、同じページ内の複数箇所で同じ値を指定しても問題ありません。また、値は半角スペースで区切って同時に複数指定することもできます(id属性の値には半角スペースは入れられません)。id属性の値と同様、class属性の値も大文字と小文字は区別されますので注意してください。

▶▶ title属性

要素に対して補足的な情報を加えたい場合に指定します。一般に、この属性に指定した値はツールチップとして(マウスカーソルを載せたときに)表示されます。

▶▶ lang属性

要素内容の言語(日本語・英語など)を示す場合に使用します。値には、日本語なら「ja」、英語なら「en」のように言語コードを指定します。

LECTURE 4-4

背景を指定する(1)

では、ここまでに覚えた要素に対して、さっそくCSSを指定してみることにしましょう。といっても、HTMLの枠組みとなっている要素の中で表示指定の対象にできる要素はbody要素くらい※6ですので、まずはbody要素(ページ全体)の背景を指定してみます。
また、「Lecture 4-2　CSSの組み込み方」ではstyle要素について学習しましたので、CSSはここではstyle要素の内容として書き込みます(そのあとは基本的にlink要素を使用します)。

▶▶ background-colorプロパティ

はじめに、Chapter 2ですでに行ったように、背景色を指定してみます。
ただし、Chapter 2では`background`というプロパティを使用しましたが、ここではそれとは少し違う`background-color`という名前のプロパティを使用します(backgroundプロパティについてはChapter 7で詳しく解説しますが、`background`は背景色以外にも色々と指定できるプロパティです)。
background-colorプロパティに指定できる値は、次の通りです。

background-colorに指定できる値

- **色**
  色の書式に従って任意の背景色を指定します。
- **`transparent`**
  背景色を透明にします。

本書では、プロパティの値のうち日本語で書いてある部分はその意味するものに置き換えた値で指定することを意味し、半角のアルファベットや記号で書いてある部分はキーワード(または書式の一部)としてそのまま指定することを意味しています。これ以降、各プロパティに対して指定できる値については、そのルールで統一して書いてありますので覚えておいてください。

たとえば、`background-color`に指定できる値のうち、「色」については`black`、`white`、`red`のような一般的な英語の色名に置き換えて指定します(色を指定するための詳しい書式についてはChapter 5で詳しく解説します)。「`transparent`」はキーワードですので、「`background-`

※6：正確に言えば、html要素に対してもCSSで表示指定をおこなうことができます。

color: transparent;」のようにそのまま指定することで、背景が透明になります。

なお、background-colorはHTMLのすべての要素に指定でき、初期値はtransparent、つまり透明です。Chapter 2で見出しや本文に特に背景色がつかずに、body要素の背景色が透けて見えていたのは、見出しや本文の背景色が初期値の transparent になっているためです。ただし、body要素だけは特別で、初期状態ではブラウザで設定されている色が表示されるようになっています。

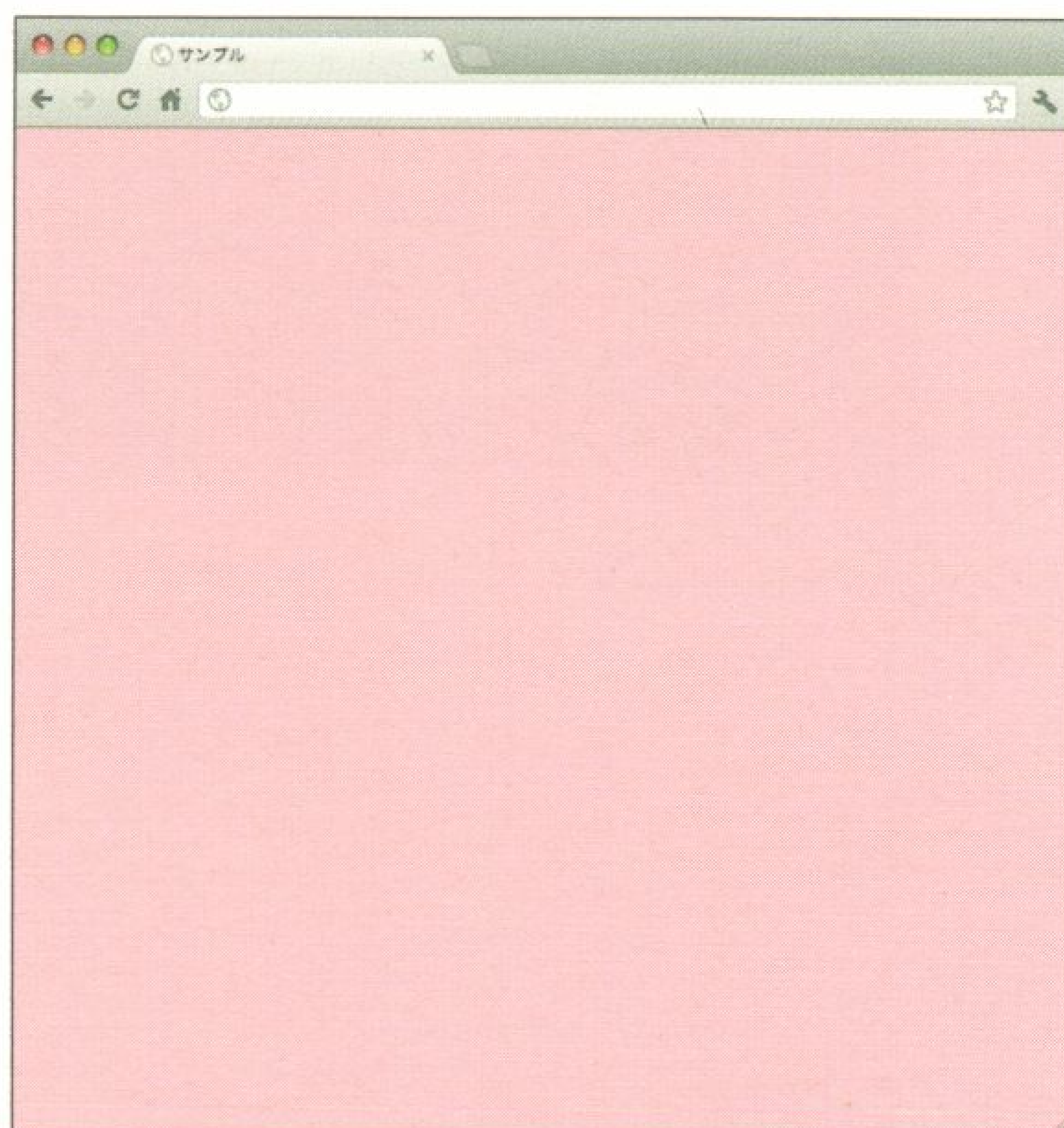
それでは、さっそくHTMLの枠組みのbody要素に対して背景色を指定してみましょう。次のサンプルでは pink を指定しています。

sample/chapter-04/lecture-4-4/01-background-color.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>サンプル</title>
<style>
body { background-color: pink; }
</style>
</head>
<body>

</body>
</html>
```

body要素の背景色を「pink」に指定している例



body要素の背景色として「pink」を指定したときの表示例

テキストエディタでそれ以外の色(black・white・gray・red・blue・yellow・green)に書き換えて保存し、ブラウザで再表示させて色が変化することを確認してみてください。



「pink」以外の色を指定した場合の表示例

▶▶ background-imageプロパティ

さて、次はbody要素の背景に画像を表示させてみましょう。
背景画像を表示させるには、background-imageプロパティを使用します。指定できる値は次の通りです。

background-imageに指定できる値

- **url(画像のアドレス)**
  画像のアドレスを指定して、その画像を背景に表示させます(書式の先頭にあるurlという部分との混乱を避ける目的で、あえて「画像のURL」とは書かずに「画像のアドレス」と表記しています)。画像のアドレス部分は、"" または '' で囲っても問題ありません。

- **none**
  背景に画像を表示しない状態にします。

ではさっそくbody要素に背景画像を指定してみましょう。
画像ファイルは、左のサンプルファイルと同じフォルダ内の「images」というフォルダの中に入っている「photo.jpg」を使用します。

sample/chapter-04/lecture-4-4/02-background-image.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>サンプル</title>
<style>
body { background-image: url(images/photo.jpg); }
</style>
</head>
<body>

</body>
</html>
```

body要素の背景として「images/photo.jpg」を指定している例

このサンプルを表示させると、次ページの図の左側のように表示されます。
しかし、ウィンドウを広げてみると、右側のように画像がタイル状に繰り返されていることが分かります。この繰り返しを制御するには、次に紹介するbackground-repeatプロパティを使用します。



背景全体に画像が表示された



画像がタイル状に繰り返されている

COLUMN

背景画像のURLについて

background-imageプロパティの値は url(画像のアドレス) の書式で指定しますが、この「画像のアドレス」の部分には絶対URLと相対URLの両方が指定できます。絶対URLとは、「http://www.example.com/images/photo.jpg」のようなブラウザのアドレスバーなどに表示される形式です。相対URLは、同じディスク上にあるファイルを相対的な位置で示す形式で、下の階層にあるファイルやフォルダの名前は / で区切って後ろに続けて示し、上の階層の場合は階層の数だけ前に ../ をつけて示す形式です。たとえば、2つ上の階層にある imagesフォルダの中の photo.jpg を指定する場合は ../../images/photo.jpgとなります。

▶▶ background-repeatプロパティ

background-repeatプロパティは、背景画像をどのように繰り返して表示させるか、または繰り返さないかを指定するプロパティです。次の値が指定できます。

background-repeatに指定できる値

- `repeat`
  背景画像を縦横に繰り返して(タイル状に並べて)表示させます。

- `no-repeat`
  背景画像を繰り返さずに、1つだけ表示させます。

- `repeat-x`
  背景画像を横にのみ繰り返して表示させます。

- `repeat-y`
  背景画像を縦にのみ繰り返して表示させます。

さきほどのサンプルで使用した背景画像は大きすぎて繰り返しの状態を確認しにくいので、ここでは右のような小さめの画像を使用します。



次のサンプルで表示させる背景画像。縦150ピクセル×横150ピクセル

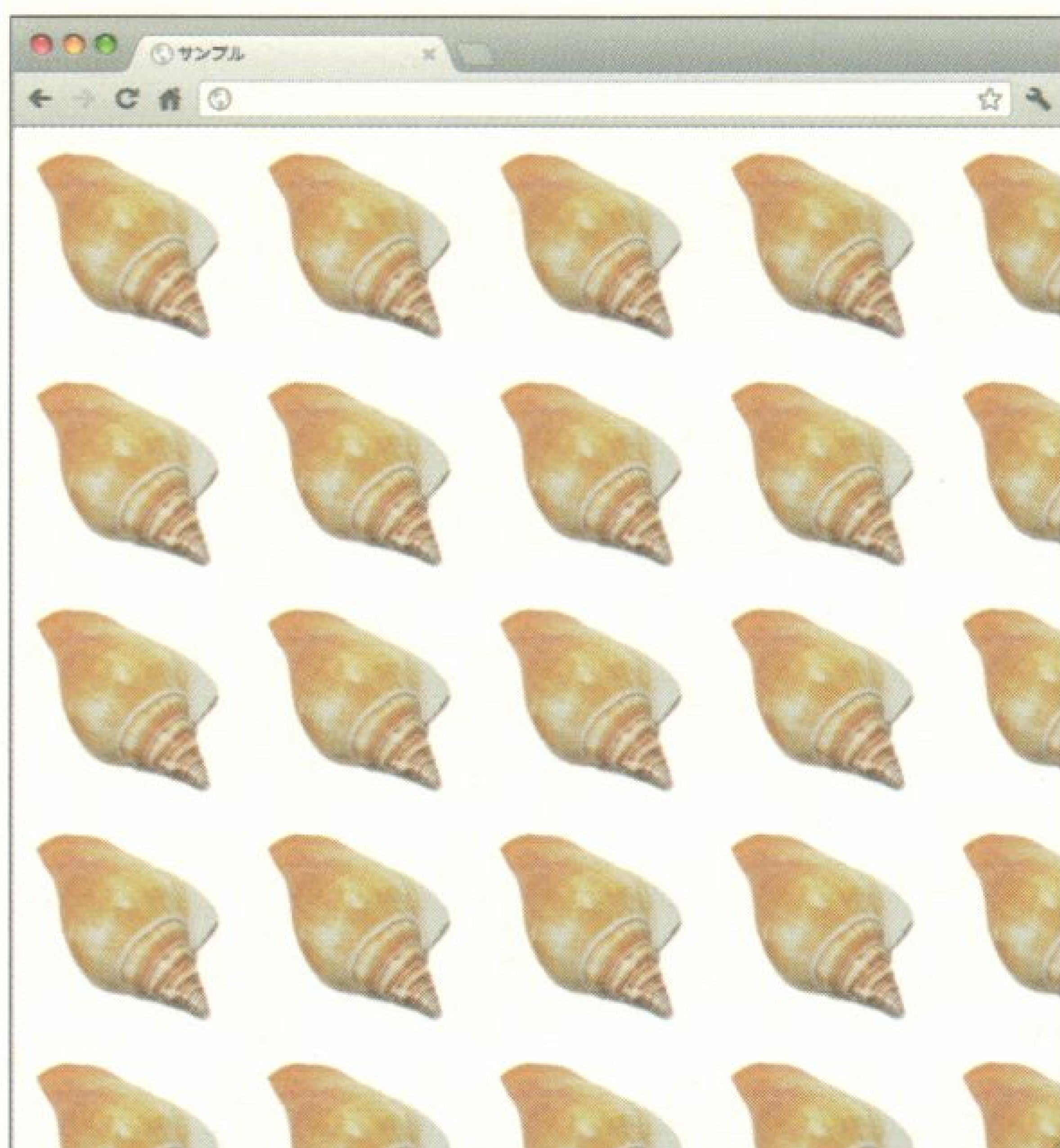
では、この背景画像を指定し、さらにbackground-repeatプロパティも指定してみましょう。以下の例では、値は初期値の「`repeat`」になっていますが、テキストエディタでそれ以外の値にも変更して次ページの図のように並び方が変化することを確認してください。

sample/chapter-04/lecture-4-4/03-background-repeat.html

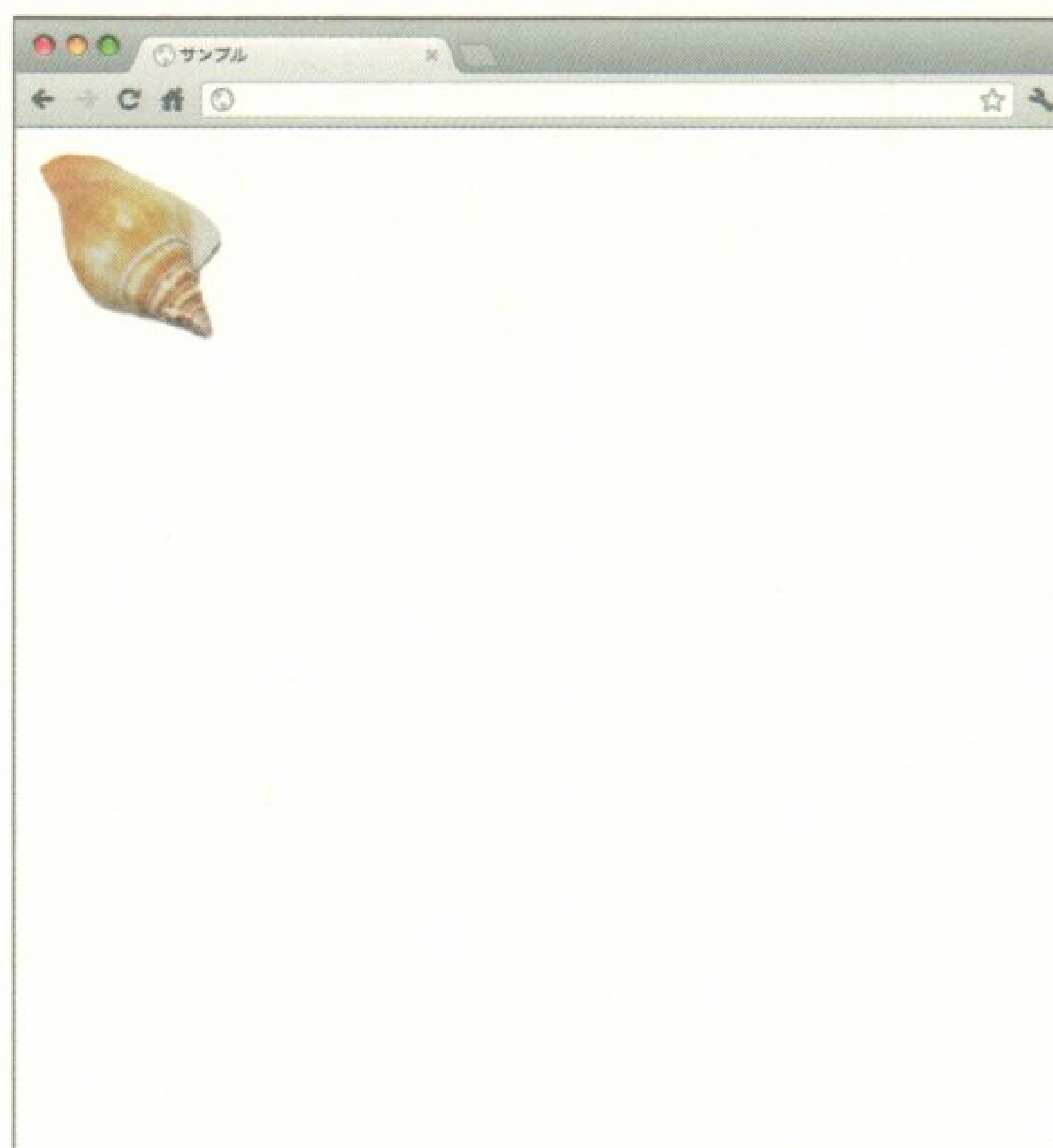
```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>サンプル</title>
<style>
body {
  background-image: url(images/shell.jpg);
  background-repeat: repeat;
}
</style>
</head>
<body>

</body>
</html>
```

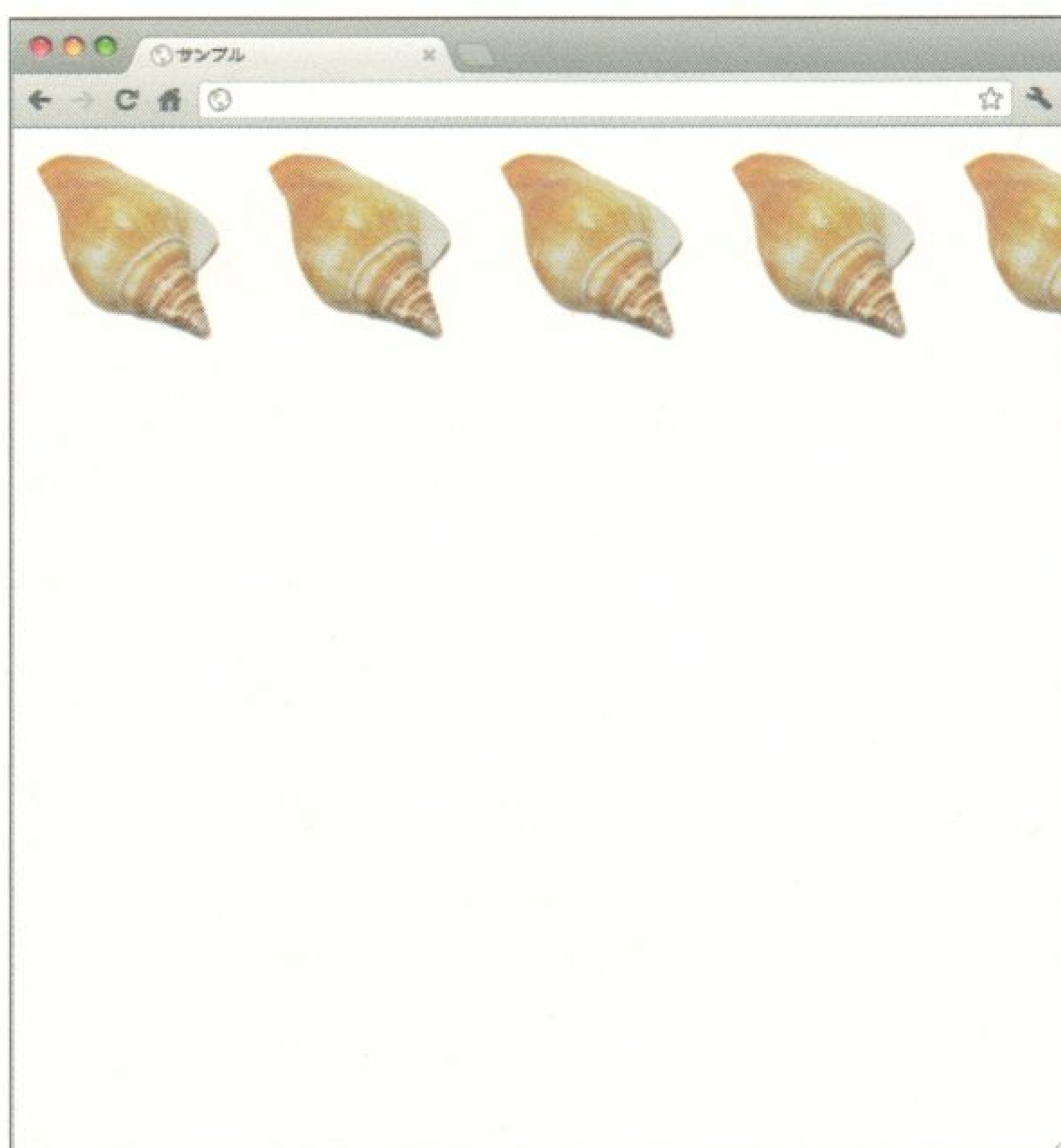
背景画像を指定し、縦横に繰り返して表示するように指定している例



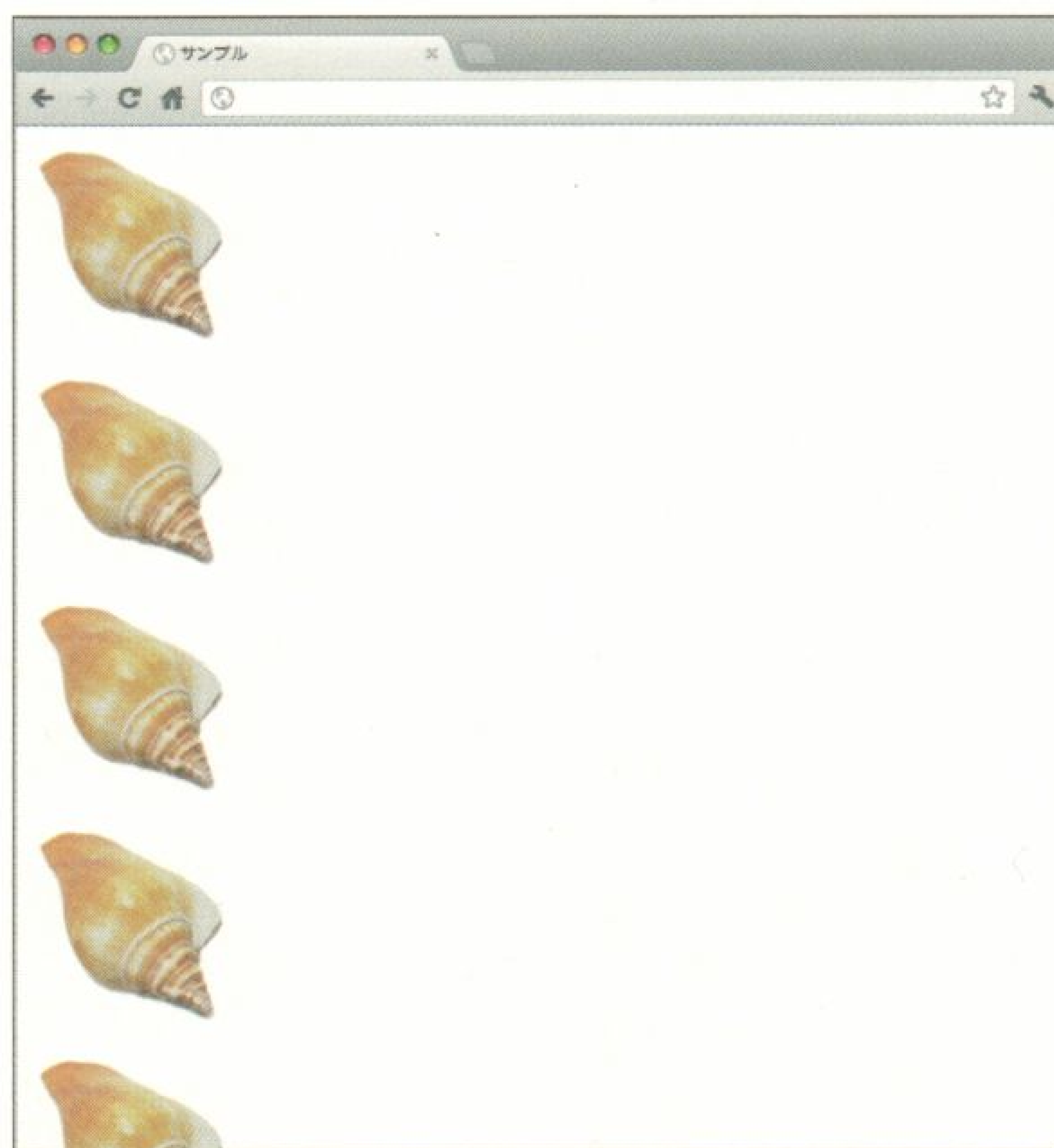
値がrepeatのときの表示



値がno-repeatのときの表示



値がrepeat-xのときの表示



値がrepeat-yのときの表示

背景に関連するプロパティはこれら以外にもありますが、残りのプロパティに関してはChapter 7で（もう少しほかの要素とプロパティを覚えた段階で）紹介します。

CHAPTER 5

テキスト

ページ全体の枠組み部分を覚えたら、次はその内容です。Chapter 5では、Webページの主要な構成要素であるテキストに関連する要素とプロパティについて一通り解説します。また、色を指定するためのさまざまな書式と関連プロパティについてもここで紹介します。

LECTURE 5-1

テキスト関連の要素

HTML5の要素は、特徴から8つに分類されています。ここでは、そのうちテキスト関連の要素について、意味合いと使い方を紹介します。

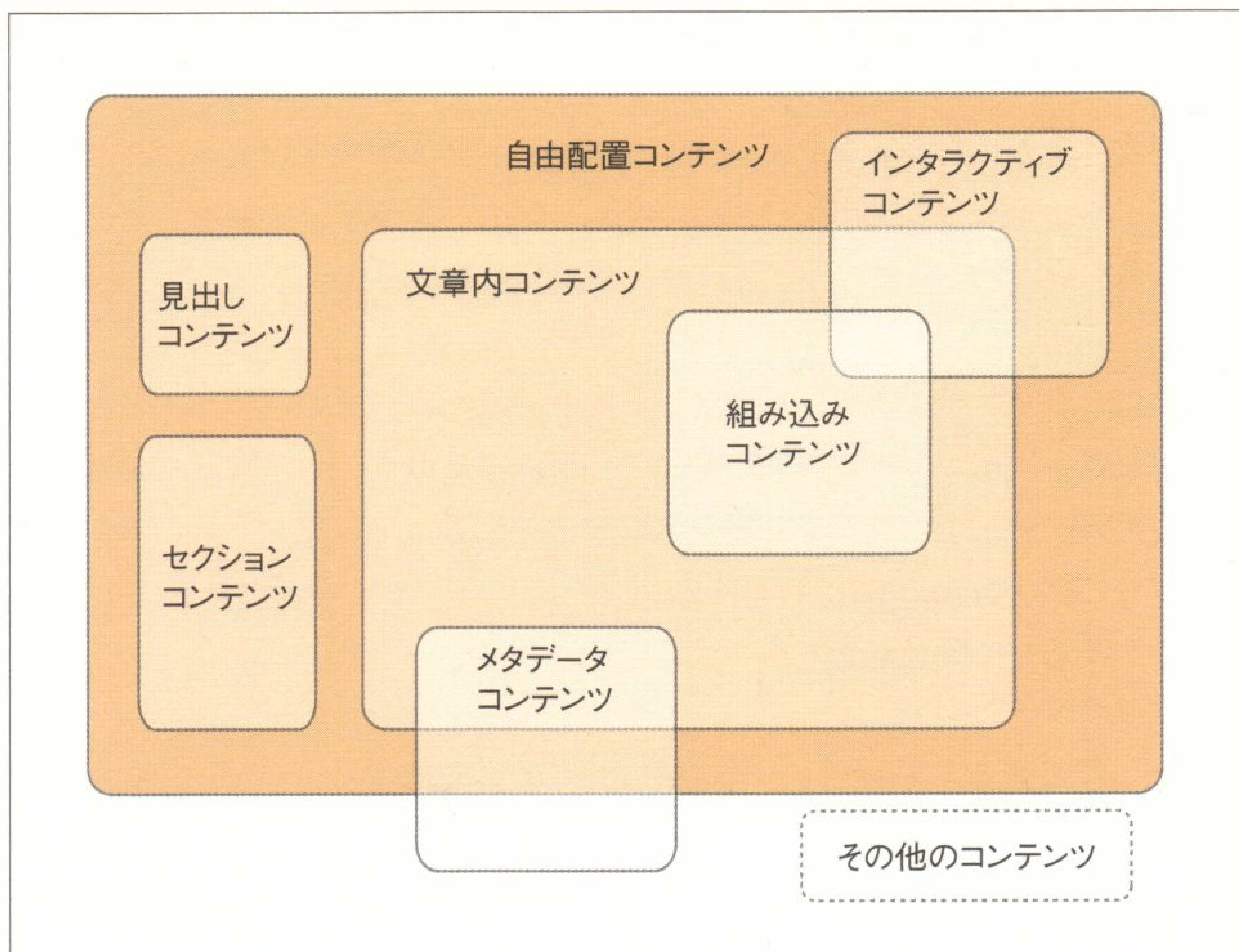
要素の分類 【HTML5改】

HTML5には約100種類の要素があり、それらは大きく次の8種類のカテゴリーに分類されます。具体的にどの要素がどのカテゴリーに該当するかは Appendixの「HTML5の要素の分類」に掲載してありますが、今の段階では「このようなカテゴリーがある」ということだけ覚えておけばOKです。

1. 自由配置コンテンツ(Flow content)
2. 見出しコンテンツ(Heading content)
3. セクションコンテンツ(Sectioning content)
4. 文章内コンテンツ(Phrasing content)
5. 組み込みコンテンツ(Embedded content)
6. インタラクティブコンテンツ(Interactive content)
7. メタデータコンテンツ(Metadata content)
8. その他のコンテンツ

これら8種類のカテゴリー同士の関係は次ページの図のようになっており、多くの要素は複数のカテゴリーに含まれます(逆にどのカテゴリーにも含まれない要素もあります)。

自由配置コンテンツとは、ある特定の要素の内部にしか配置できないといった制限がなく、body要素の内部であれば基本的にどこにでも配置できる要素という意味で、ほとんどの要素はまずこのカテゴリーに含まれています。そして、それらのほとんどは自由配置コンテンツでありつつ、他のカテゴリーにも含まれます。



HTML5の要素のカテゴリー。多くの要素は複数のカテゴリーに該当するが、どれにも該当しない要素もある

HTML5では、「どの要素をどこに配置できるか」「どの要素にはどの要素を入れられるか」といった情報を示す際にこのカテゴリーを使用します。詳細はAppendixの「HTML5の要素の配置のルール」に掲載してありますので、本書を読み進めながら必要に応じて参照してください。

さて、まだ要素もほとんど覚えていない段階ですので、8種類のカテゴリーについてはここで覚える必要はありません。しかし、HTML5よりも前のバージョンのHTMLで使用されていた「おおまかな分類方法」については、覚えておいた方が良いでしょう。なぜなら、その分類方法はHTML4.01やXHTML1.0、さらにはCSS2.1で今でも主要な（単純明快で分かりやすい）概念として使われているからです。

その考え方とは、すべての要素を大きく「**ブロックレベル要素**」「**インライン要素**」「**その他の要素**」の3種類に分けるというものです。簡単に言えば、ブロックレベル要素とは、1つのまとまった単位となっているひとかたまりの（ページ内の1ブロックを構成するような）テキストのことを示します。たとえば、見出しや段落（Chapter 2で登場したh1要素とp要素）はブロックレベル要素の分かりやすい例です。

それに対してインライン（inline=行内）要素とは、ブロックレベル要素の中に入っているテキスト（文章）の一部分となるような要素のことを言います（ただし、必ずしもテキストだけではなく、文章の一部に入ることもある画像などもインライン要素に含まれます）。HTML5における「文章内コ

ンテンツ」は、この「インライン要素」とほぼ同様の分類にあたります。

なお、インライン要素や文章内コンテンツには、ブロックレベル要素内で特にタグがつけられていない普通のテキストも含まれます。つまり、「要素内容としてインライン要素を入れることができます」と書いてある場合は、要素内容として普通のテキストも入れられるということを意味します。

ブロックレベル要素は1つのまとまりとして独立したものですので、その前後が他の要素のテキストと同じ行のままつながることはありません。基本的には、ブロックレベル要素の前後は改行された状態となります。それに対してインライン要素は文章の一部として使用しますので、その前後は同じ行のままつながって表示されます(これらはCSSを指定していない状態での話で、CSSを使用すれば表示はどのようにでも変更できます)。

ちなみに、ブロックレベル要素とインライン要素のどちらにも該当しない“その他の要素”には、Chapter 4で学習したHTMLの枠組みとして使用される要素や、たとえば表の内部でのみ使用可能な要素のように限定された特定の範囲内でしか使用できない要素などが含まれています。

それでは、HTML5の複雑な分類ではなく、まずはこのブロックレベル要素とインライン要素というシンプルな分類で、HTML5のテキスト関連要素を見ていきましょう。

▶▶従来のブロックレベル要素に該当する要素【HTML5改】

右の表に示したのが、テキスト関連で従来のブロックレベル要素に該当する要素です。ただし、従来のブロックレベル要素に該当する要素はこれで全部というわけではありません。ここで紹介している要素は、あくまでテキスト部分にタグをつける際の基本となるブロックレベル要素です。

要素名	意味
h1～h6	見出し
hgroup	見出しのグループ化
p	段落
blockquote	引用文
pre	整形済みテキスト
div	汎用ブロックレベル要素

HTML5のテキスト関連要素のうち、従来のブロックレベル要素に該当する要素

「h1～h6」と表記した要素は、h1・h2・h3・h4・h5・h6という6種類の要素をまとめて表したものです。Chapter 2ではh1要素だけを紹介しましたが、アルファベットの「h」は「heading」の略で、これが見出しをあらわす要素であることを示しています。それに続く1～6の数字は、見出しの階層(レベル)をあらわしており、1が大見出し(一番上の階層)で数が大きくなるほど下の階層の見出しとなります。

しかし、HTML5よりも前のHTML・XHTMLにおいては、この数字でしか階層を示せませんでしたが、HTML5では新しい別の方法も用意されています。見出しの階層を示す別の方法については、それに関連する新要素とともにChapter 7で解説します。

hgroup要素は、**見出しをグループ化する**要素です。見出しをグループ化するための専用要素ですので、要素内容として入れられるのはh1〜h6要素のいずれかだけです。

一般に新しい見出しが登場すると、そこから新しい章や節が始まると認識されます（ブラウザもそのように認識します）。しかし、1つの見出しの内容を主題と副題のように分けてh1要素とh2要素を連続して使ったような場合は、それがまとめて1つの見出しとして扱うべきものであるのか、それとも「第1章 第1節」のように2つの独立した見出しとして扱うべきものなのか、ブラウザは判断することができません。そこで、連続する2つの見出しが主題と副題である（つまり2つセットで1つの見出しである／見出しの2つの階層を表しているわけではない）ことを明確に示したい場合に、それらを<hgroup>〜</hgroup>で囲う、といった使い方をします。
hgroup要素についてもChapter 7で再度詳しく解説しますので、ここでは見出しはグループ化できるということだけを覚えてください。

```
<hgroup>
<h1>かちかち山</h1>
<h2>Win-Winが大好きな山田社長の物語</h2>
</hgroup>
```

hgroup要素の使用例

p要素は、Chapter 2で紹介したとおり、その内容が**1つの段落**（paragraph）であることを示す要素です。内容として入れられるのは**インライン要素**に該当する要素だけで、ブロックレベル要素は入れることができない点に注意してください。p要素は見出し関連要素と共にもっとも頻繁に使われる要素の1つですので、しっかりと覚えておきましょう。

blockquote要素は、その部分が**引用文**であることを示すブロックレベル要素です。もし、ブロックレベル要素としてではなく、文章内の一部分（インライン要素）として引用文を含める場合は、後述の**q要素**を使用します。blockquote要素の内容には、ブロックレベル要素でもインライン要素でも自由に入れられます。**cite属性**を指定して引用先のURLを示すこともできます。この要素は、ブログの記事で別の記事を引用する場合などによく使われます（一般的なブログサービスでは、このタグをつけると引用文向けに用意された表示指定が適用されるようになっています）。

```
<blockquote cite="http://example.com/about.html">
<p> 〜 引用文 〜 </p>
</blockquote>
```

blockquote要素の使用例

pre要素は、その内容が整形済みのテキスト（preformatted text）であることを示す要素です。一般的なブラウザでは半角スペースや改行などが入力した状態のままで等幅のフォントで表示されます（具体的な表示方法はもちろんCSSで変更可能です）。ASCIIアートを表示させるような場合にも使用できますが、ソースコードを表示させる場合などに多く使用されます。要素内容として入れられるのはインライン要素に該当する要素だけで、ブロックレベル要素は入れられない点に注意してください。

div要素のdivはdivisionの略で、ただその範囲がブロックレベル要素であることだけを示します。基本的には、ブロックレベル要素の中で他にふさわしい要素がない場合に使用しますが、複数のブロックレベル要素をグループ化する際などに多く使用されます。div要素の内容には、ブロックレベル要素でもインライン要素でも自由に入れられます。

▶▶ 従来のインライン要素に該当する要素（1） 【HTML5改】

次は、テキスト関連で従来のインライン要素に該当する要素です。テキスト関連のインライン要素は数が多いので、比較的多く使用されるものと、それほど多くは使用されないものの2つに分けて紹介します。

なお、各要素の解説部分では触れませんが、インライン要素の場合、特別な例外を除いて要素内容として入れられるのはインライン要素だけです（つまり内容としてブロックレベル要素は入れられません）。では、はじめに比較的多く使用されるインライン要素から見ていきましょう。

要素名	意味
br	改行
em	強調部分
strong	重要部分（重要性）
q	引用文
cite	作品名
code	ソースコード
small	一般に小さな文字で掲載される部分
span	汎用インライン要素

HTML5のテキスト関連要素のうち、比較的多く使用されるインライン要素

Chapter 3で説明したとおり、HTMLのソースコードの中で改行を入れても、ブラウザで表示させたときには半角スペースに置き換えられてしまい、その位置では改行はしません。ブラウザで表示させたときに改行させるためには、改行させたい位置にbr要素を配置します。たとえば住所と名前を掲載する場合に、住所と名前の間に改行を入れるような用途で使用します。

br要素は単純にその位置で改行させるためだけに使用される要素ですので、要素内容と終了タグのない空要素として定義されています。なお、br要素のbrは、改行をあらわす英語「line break」の「br」です。

```
<p>
〒012-3456<br>
北海道札幌市鳥獣保護区1-2-3<br>
大藤 幹
</p>
```

br要素の使用例

em要素は、その部分が強調されていることを示す要素です。emは、強調という意味の英単語「emphasis」の略です。一般的なブラウザでは斜体で表示されますが、表示方法はCSSで自由に変更できます。

strong要素は元々はem要素よりもさらに強い強調部分を示す要素でしたが、HTML5からはその部分が重要であることを示す要素に変更されました。一般的なブラウザでは太字で表示されますが、表示方法はCSSで自由に変更できます。

em要素が示す「強調」と、strong要素が示す「重要性」は一見すると同じことのように感じますが、「強調」はどこを強調するかによって文章の意味が変わってしまうという点が異なります。たとえば、「僕は彼女が好きです」のようにタグを付けた場合、ほかの人はどうか分からないが“僕は”好きだという意味になります。しかし、「僕は彼女が好きです」になると、僕はほかの人ではなく“彼女が”好きだという意味になります。また、強調するということは、Webページの内容を音声で読み上げる場合のアクセントにも影響を与えます。strong要素が示す「重要性」は、文章の意味を変えてしまうことはないという点で、em要素が示す「強調」とは異なります。

q要素は、その部分が引用文(quotation)であることを示すインライン要素です。同じ引用文をあらわすblockquote要素のインライン版です。blockquote要素と同様に、引用先のURLを示す**cite属性**も指定できます。一般に、日本語で文章中に引用文を含む場合にはその部分を「 」で囲いますが、そのような記号はCSSで表示させることになっています(具体的な指定方法はChapter 10で解説します)。要素内容のテキストとして、「 」やその他の引用符などは含めないように注意してください。

ただし、インラインの引用文に関しては、q要素のタグをつけて表現するほかに、あえてq要素のタグをつけずに自分で「 」やその他の引用符をつけて示すことも認められています。日本語の場合は、原稿のテキストの一部として「 」を含んでいるのが一般的ですので、特にq要素は使わないで元の原稿そのままにしておいてもまったく問題はありません。

```
<p>
アイヌ民族最後の狩人と呼ばれる姉崎氏によると<q>空のペットボトルを押して出る音をクマは
嫌います</q>とのことだ。
</p>
```

```
<p>
アイヌ民族最後の狩人と呼ばれる姉崎氏によると「空のペットボトルを押して出る音をクマは嫌い
ます」とのことだ。
</p>
```

インラインの引用文は、引用符をとって<q>〜</q>で囲っても、q要素にせずに原稿そのままにしておいてもOK

cite要素は、その部分が(引用文や話題として触れた)**作品のタイトル**であることを示す要素です。この要素は、人名に対しては使用できません。書籍や論文、詩、エッセイ、映画、演劇、テレビ番組、歌、楽譜、芸術作品、ゲームなどのタイトルに対して使用されます。

code要素は、その部分が**ソースコード**であることを示す要素です。ソースコード中の改行やインデントなどをそのまま表示させたい場合には、ブロックレベル要素であるpre要素の内部でcode要素を使用してください。

```
<pre><code>body {
    background-image: url(photo.jpg);
    background-repeat: repeat;
}</code></pre>
```

CSSのソースコードをコンテンツとして掲載する場合のcode要素の使用例。ソースコード中の改行やインデントをそのまま再現したい場合は、pre要素の内部で使用する

small要素は、**一般に小さな文字で掲載される部分**(著作権表示・注意書き・免責事項のような付帯情報的な部分)であることを示す要素です。元々はフォントサイズを小さくして表示させるための要素でしたが、HTML5からそのような意味に変更されました。

```
<p>
<small>Copyright &copy; 2012 ○○○, Inc. All rights reserved.</small>
</p>
```

small要素の使用例。「©」部分は「©」のように表示される

span要素は、div要素のインライン版で、その範囲が**インライン要素であることだけ**を示します。ほかのインライン要素の中にふさわしいものがないような場合に使用します。

▶▶ 従来のインライン要素に該当する要素 (2) ｜HTML5改｜

次に、あまり多くは使用されないインライン要素を見ていきましょう。

要素名	意味
abbr	略語
dfn	定義対象の用語
sup	上付き文字
sub	下付き文字
i	一般にイタリック体で掲載される部分
b	一般に太字で掲載される部分
mark	注目してほしい部分

HTML5のテキスト関連要素のうち、それほど頻繁には使用されないインライン要素

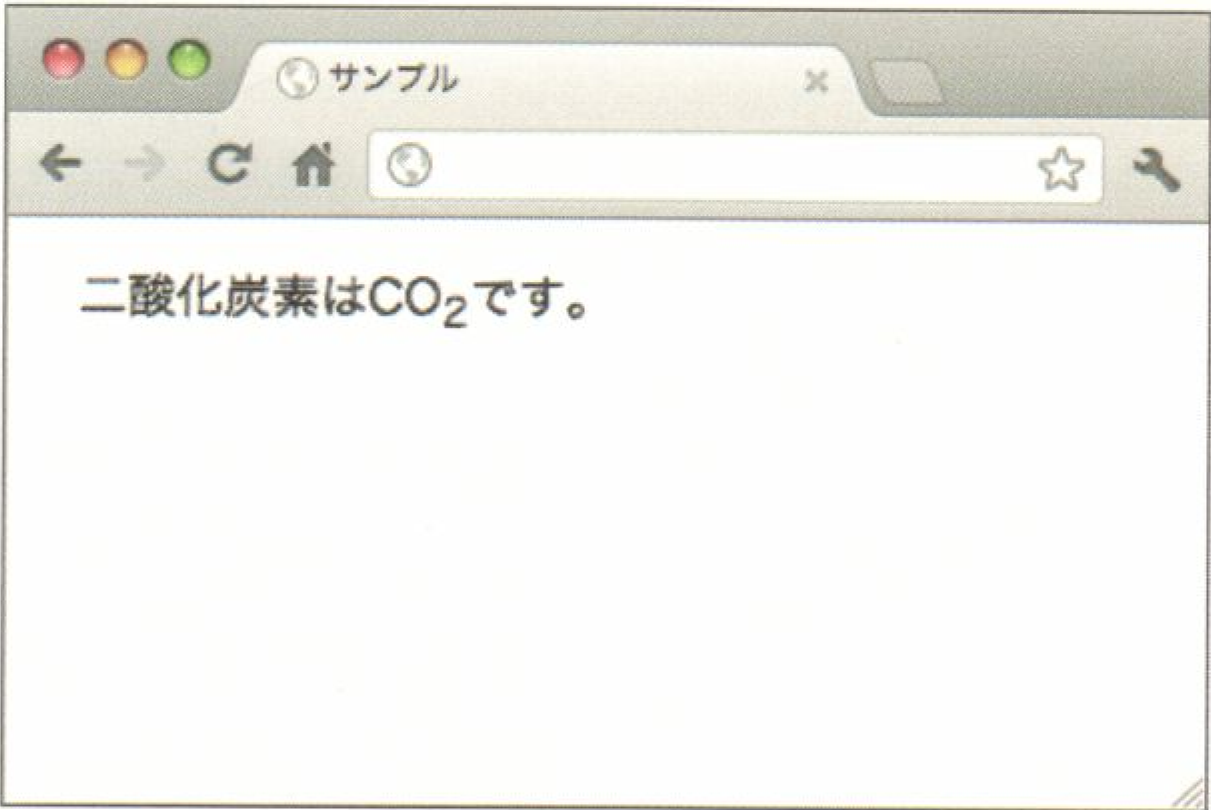
abbr要素は、その部分が**略語**(abbreviation)であることを示す要素です。その略語が何の略であるかを示したい場合は、title属性の値として「省略していない状態の言葉」を指定することができます。

dfn要素は、その部分が**定義**(defining)対象の用語であることを示す要素です。「○○○とは、△△△△△のことです」といった何かを定義する文章の、○○○の部分に対して使用します。

sup要素はその部分が**上付き文字**(superscript)であることを、**sub要素**はその部分が**下付き文字**(subscript)であることを示す要素です。一般的なブラウザでは、それぞれ上付き文字・下付き文字として表示されます。

```
<p>
二酸化炭素はCO<sub>2</sub>です。
</p>
```

sub要素の使用例



sub要素の表示例

i要素は、もともとは単純に要素内容をイタリック体(italic)で表示させるための要素でした。しかしHTML5からは、生物学における「学名」や、英語における「船の名前」や「外国語」などのように、**慣習として通常はイタリックで表記する部分**に使用する要素となりました。

b要素は、もともとは単純に要素内容を太字（bold）で表示させるための要素でした。しかしHTML5からは、文書中の「キーワード」を目立たせたい場合や、製品レビュー中の「製品名」などのような一般に太字で掲載されるような部分に使用する要素となりました。このタグをつけたからといって、特に重要性などを表すわけではない点に注意してください。

mark要素は、その部分がユーザーに注目してほしい部分であることを示す要素です。たとえば、ある引用文があって、元のテキストでは特にタグはつけられていないけれども読んでいる人に注目してほしい部分を示す場合や、検索結果の一覧表示で検索に使用されたキーワード部分をハイライト表示したい場合などに使用されます。

▶▶ リンク 【HTML5改】

さて次は、HTML5より前のHTMLではインライン要素に分類されていた、リンクを作成するための要素であるa要素を紹介しましょう。ちなみにa要素の「a」は、「（hypertext） anchor」の「a」です。
テキストの一部をリンクにするには、その範囲を**a要素**のタグで囲って、リンク先のURLを**href属性**（hrefはhypertext referenceの略）で指定するだけです。たとえば、次のテキストの「サンプル」という部分を「http://www.example.com/」にリンクさせたい場合は、次のように記述します。

```
<p>
わかりやすい<a href="http://www.example.com/">サンプル</a>もあります。
</p>
```

「サンプル」というテキストをリンクにするときの記述例

このa要素は、HTML5より前はインライン要素として分類されていたために、内容として入れることができるのはインライン要素だけでした。しかし、HTML5からは、そのa要素の親要素に入れられる要素であれば、どの要素でも入れられるように仕様変更されています。つまり、a要素を抜きで考えてそこにあっても問題のない要素であれば、どの要素でも～で囲ってリンクにできるということです。したがって、h1～h6要素やdiv要素、ul要素といったブロックレベル要素をa要素の中に入れても、HTML5では文法エラーにはなりません。
ただし、そこには例外もあります。a要素の内部には、ほかのa要素およびインタラクティブコンテンツに分類される要素（フォームで使用される部品など）は一切入れられないことになっています。これはつまり、リンクの中に別のリンクが含まれていたり、リンクの中にテキスト入力欄やメニューやボタンなどがあってはいけないということを意味しています。

a要素には、href属性のほかに**target属性**も指定できます。target属性を使用すると、リンク先を新しいウィンドウやタブに表示させることなどが可能になります。

a要素に指定できる属性

・**href="リンク先のURL"**
この属性を指定することでa要素はリンクになります。
値にはリンク先のURLを指定します。

・**target="リンク先を表示させるウィンドウまたはタブ名"**
リンク先を表示させるウィンドウまたはタブを指定する場合に使用します。
値にはウィンドウまたはタブの名前が指定でき、指定した名前のものがすでにあればそこに表示し、なければ新しいウィンドウまたはタブに表示されます。また、値にはアンダースコアではじまる特別なキーワードも指定できます。
「_blank」を指定すると、リンク先は新しいウィンドウまたはタブに表示されます。
「_self」を指定すると、リンク先は現在のウィンドウまたはタブに表示されます。
「_parent」はChapter 10で登場するiframe要素を使っていて親ページがある場合に使用し、親となっているウィンドウまたはタブがある場合はそこへ、なければ現在のウィンドウまたはタブに表示させます。

COLUMN

ページ内の特定の場所にリンクする

href属性で指定するURLの最後に半角の「#」をつけ、そのあとにid属性で指定してある値を加えることで、指定したページ内のそのid属性が指定されている要素の位置へとリンクさせることができます(リンク先がある程度長いページの場合は、その要素が表示されるところまでスクロールした状態で表示されます)。

リンク元のソースコード

```
<p>
<a href="http://www.example.com/index.html#section2">第2節</a>へ。
</p>
```

リンク先(http://www.example.com/index.html)のソースコード

```
・・・
<h1 id="section2">
第2節 ふざけたサンプルの本は信用するな
</h1>
・・・
```

ページ内の特定の要素の位置にリンクさせたい場合は、リンク元のURLの最後に「#」をつけ、リンク先の要素のid属性の値を指定する

ルビ 【HTML5新】

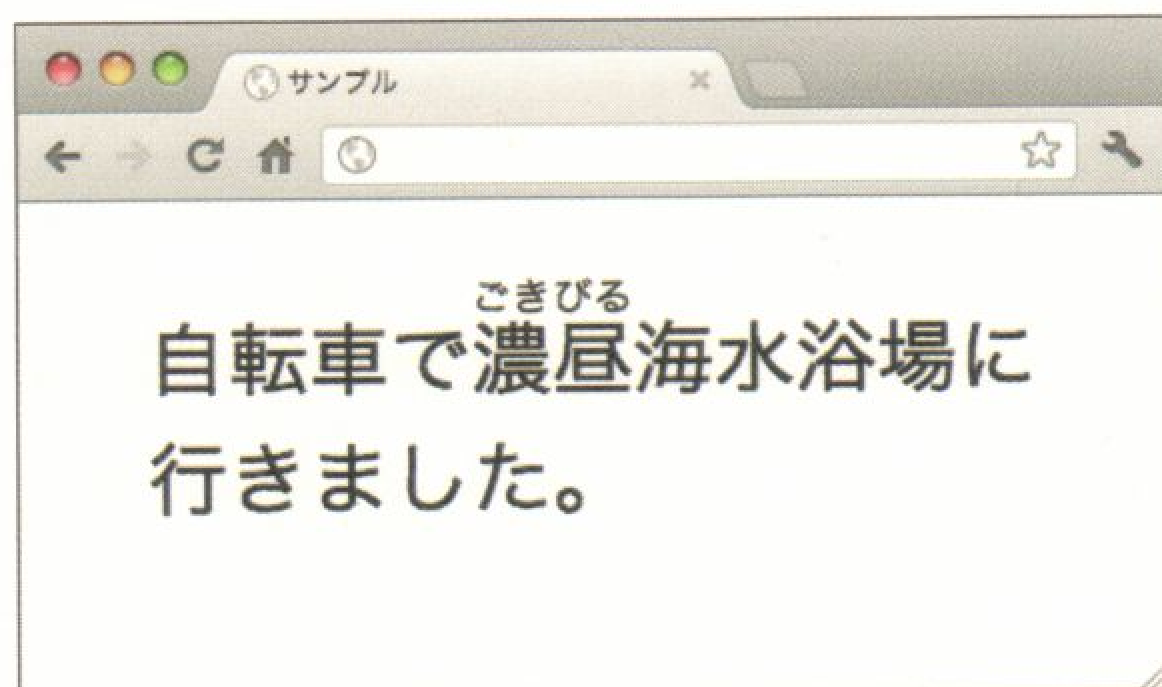
インライン要素の最後は、構造がちょっと複雑な**ruby要素**です。rubyとは日本語のルビのことで、この要素を使うことで任意のテキストに**ふりがな（ルビ）をふる**ことができます。

ある部分に単純にふりがなをつけるだけなら、ふりがなをふるテキストを<ruby>〜</ruby>で囲い、終了タグ</ruby>の直前に**rt要素**の内容としてふりがなのテキストを入るだけでOKです。ちなみに、「rt」は「ruby text」の略です。

sample/chapter-05/lecture-5-1/01.html

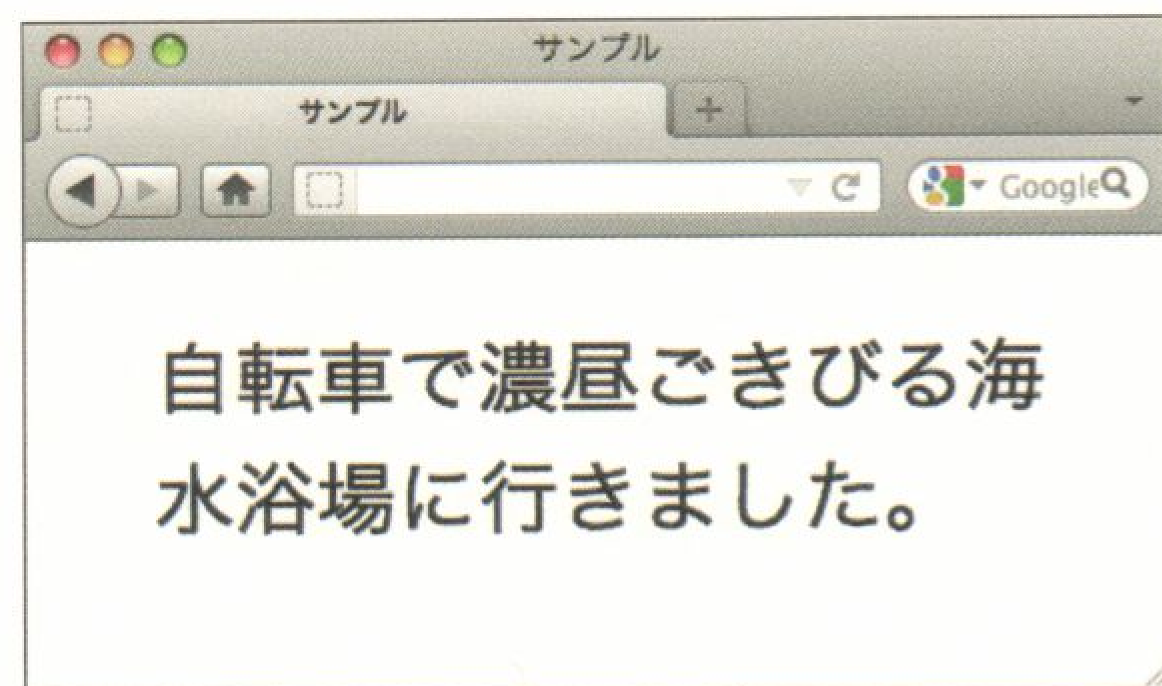
```
<p>
自転車で<ruby>濃昼<rt>ごきびる</rt></ruby>海水浴場に行きました。
</p>
```

テキストの上に表示させるふりがなはrt要素に入れて、ふりがなをふるテキストの直後に配置。それら全体をruby要素にすると、ふりがなが表示される



上のソースコードの表示例

しかし、現時点ではすべてのブラウザがruby要素に対応しているわけではありません（Google Chrome、Safari、IEは対応しています）。たとえば、Firefox 11はruby要素に未対応で、上のソースコードは次のように表示されます。



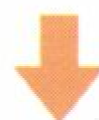
上のソースコードをFirefox 11で表示させたところ。ルビも普通のテキストとして表示される

これでは未対応ブラウザでは、内容によっては意味不明のテキストになってしまうこともあるかもしれません。そこで未対応のブラウザで見たときには「濃昼ごきびる」ではなく「濃昼(ごきびる)」のように**カッコつきで表示させる**こともできるようになっています。その際に使用するのが**rp要素**です(「rp」は「ruby parentheses」の略で、parenthesesは丸カッコを意味します)。

未対応のブラウザでふりがなの前後にカッコが表示されるようにするわけですから、開きカッコのrp要素はふりがなの直前に、閉じカッコのrp要素はふりがなの直後になければなりません。rp要素の要素内容としては、未対応のブラウザでふりがな(つまりrt要素)の前後に表示させたい開きカッコと閉じカッコをそれぞれ入れます。
たとえば、さきほどのソースコードを次のように変更すると、未対応のブラウザではふりがながカッコ付きで表示されるようになります。なお、ruby要素に対応しているブラウザでは、rp要素の内容は無視されて表示されません。

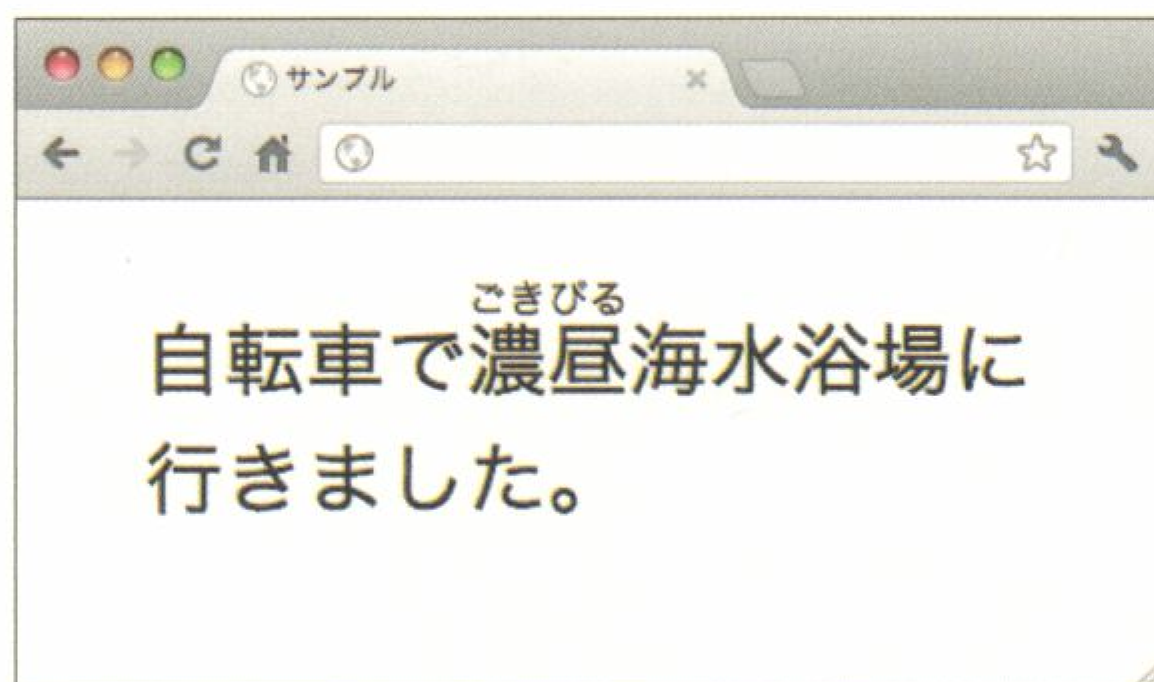
sample/chapter-05/lecture-5-1/02.html

```
<ruby>濃昼<rp>(</rp><rt>ごきびる</rt><rp>)</rp></ruby>
```

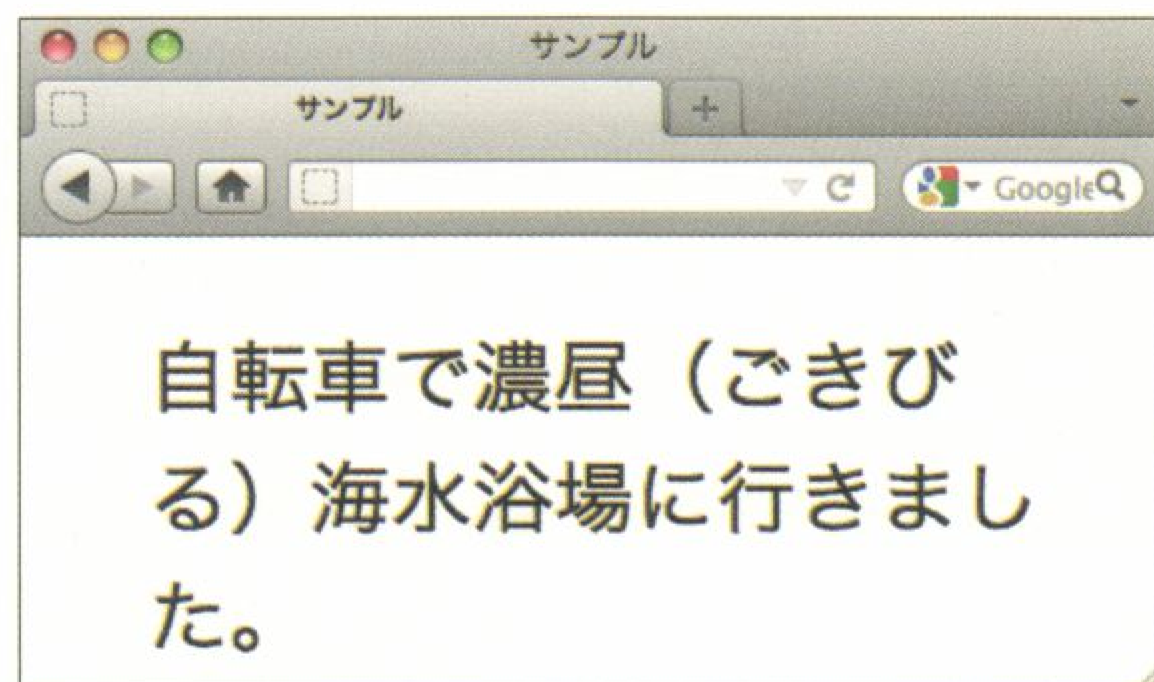


```
<ruby>
濃昼
<rp>(</rp>
<rt>ごきびる</rt>
<rp>)</rp>
</ruby>
```

ruby要素に未対応のブラウザで見たときに、rt要素がカッコ内に表示されるようにしたソースコードの例



上のソースコードのGoogle Chromeでの表示例。rp要素は無視され、()は表示されない

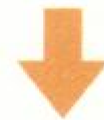


上のソースコードのFirefox 11での表示例。未対応のブラウザではテキストはそのまま表示されるので、このようになる

実はruby要素内には、「ふりがなをふるテキスト＋rp要素+rt要素+rp要素」を複数入れられる仕様となっています。つまり、次のようなふりがなのふり方も可能です。

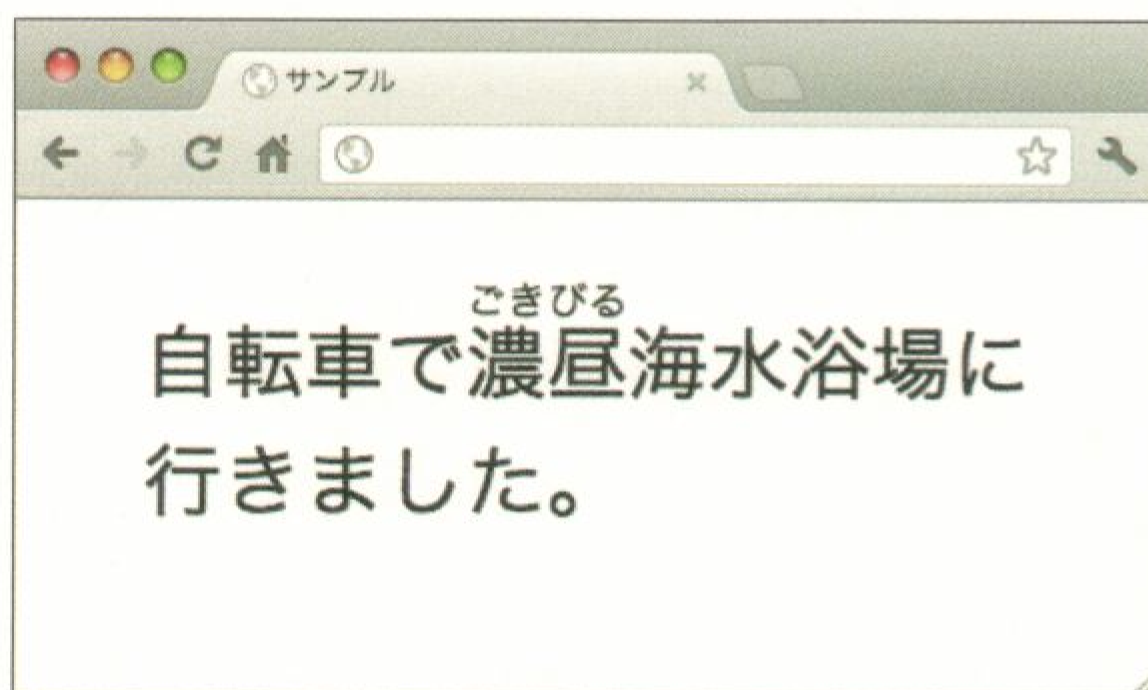
sample/chapter-05/lecture-5-1/03.html

```
<ruby>濃<rp>(</rp><rt>ご き</rt><rp>)</rp>昼<rp>(</rp><rt>び る</rt><rp>)</rp></ruby>
```

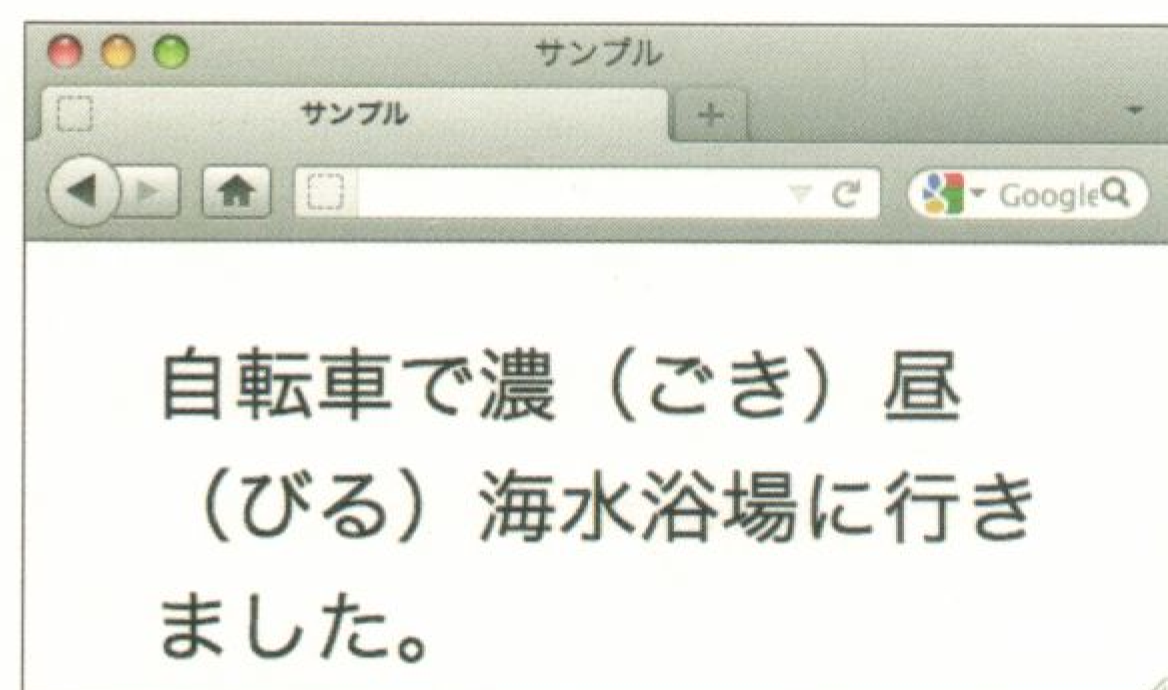


```
<ruby>
濃
<rp>(</rp>
<rt>ごき</rt>
<rp>)</rp>
昼
<rp>(</rp>
<rt>びる</rt>
<rp>)</rp>
</ruby>
```

ruby要素の中は、このように分割して細かくふりがなをふることもできる



上のソースコードのGoogle Chromeでの表示例



上のソースコードのFirefox 11での表示例

LECTURE 5-2

色の指定方法

テキスト関連の要素が一通り分かりましたので、次はそれらに対して指定できるCSSのプロパティを紹介していきます。でもその前に、今後さまざまなプロパティの値として指定することになる「色の指定方法」について、先に説明しておきましょう。

▶▶色の値の指定形式

ここまでのCSSの例では、色はblack、white、redのような色の名前で指定してきました。しかし、CSSにはほかにもいくつかの色の指定方法が用意されています。

「#ff0000」形式

記号#に続けて、RGBのRed、Green、Blueそれぞれの値を2桁ずつの16進数（計6桁）で指定する形式です。たとえば、赤のRGB値は10進数だと「Red=255、Green=0、Blue=0」です。255は2桁の16進数だとff、0は00となりますので、この形式だと赤は#ff0000となります。この形式はWebページの色指定ではもっとも一般的な方法ですので、多くのグラフィック系ソフトやツールなどはこの形式での色の値が分かるように作られています。

「#f00」形式

記号#に続けて、RGBのRed、Green、Blueそれぞれの値を1桁ずつの16進数（計3桁）で指定する形式です。この場合の1桁ずつの16進数はそのまま色の値として使用されるのではなく、各桁の数字の直後に同じ数字をもう1つずつ追加した値に変換されて使用されます。つまり、#123と指定されたら#112233、#f00と指定されたら#ff0000が指定されたことになるわけです。そのため、この形式で表せるのは、6桁のRGBの各2桁がそれぞれ同じ数字になっている色※7だけです。

「rgb（255, 0, 0）」形式

16進数を使わずに、RGBの各値をシンプルに10進数で指定する形式です。rgb( )のカッコ内に、RGB値をカンマで区切って順に入れるだけでOKです。

※7：いわゆるWebセーフカラーは、すべて3桁の16進数形式であらわすことができます。Webセーフカラーとは、256色しか表現できないパソコンで機種による色の違いを生じさせないために使用されていた色のことで、具体的にはRGBの各値を16進数の00, 33, 66, 99, cc, ffの組み合わせだけで作成した216色のことです。

「rgba(255, 0, 0, 0.5)」形式 【CSS3新】

rgb()の形式に透明度を示すalphaの値を追加した形式です。rgba()のカッコ内に、RGB値に続けて透明度も指定します。透明度は、0～1の実数であらわし、0は完全に透明で、1は完全に不透明となります。たとえば、半透明の赤はrgba(255, 0, 0, 0.5)となります。

	色の名前	#ff0000形式	#f00形式	rgb(255,0,0)形式	rgba(255,0,0,0.5)形式
	black	#000000	#000	rgb(0,0,0)	rgba(0,0,0,1)
	gray	#808080		rgb(128,128,128)	rgba(128,128,128,1)
	silver	#c0c0c0		rgb(192,192,192)	rgba(192,192,192,1)
	white	#ffffff	#fff	rgb(255,255,255)	rgba(255,255,255,1)
	fuchsia	#ff00ff	#f0f	rgb(255,0,255)	rgba(255,0,255,1)
	red	#ff0000	#f00	rgb(255,0,0)	rgba(255,0,0,1)
	yellow	#ffff00	#ff0	rgb(255,255,0)	rgba(255,255,0,1)
	lime	#00ff00	#0f0	rgb(0,255,0)	rgba(0,255,0,1)
	aqua	#00ffff	#0ff	rgb(0,255,255)	rgba(0,255,255,1)
	blue	#0000ff	#00f	rgb(0,0,255)	rgba(0,0,255,1)
	purple	#800080		rgb(128,0,128)	rgba(128,0,128,1)
	maroon	#800000		rgb(128,0,0)	rgba(128,0,0,1)
	olive	#808000		rgb(128,128,0)	rgba(128,128,0,1)
	green	#008000		rgb(0,128,0)	rgba(0,128,0,1)
	teal	#008080		rgb(0,128,128)	rgba(0,128,128,1)
	navy	#000080		rgb(0,0,128)	rgba(0,0,128,1)

CSSで使用できる基本的な色の名前と、それに対応する各形式の色の値

▶▶ 色に関連するプロパティ

ここで色に関連するプロパティを2つ紹介しておきます。1つはChapter 2でも使用したcolorプロパティで、もう1つは要素全体の透明度を指定するためのopacityプロパティです。

colorプロパティ

colorプロパティは、テキストの色を指定するためのプロパティです。このプロパティで指定している色を初期値とするプロパティがいくつかあります(たとえば、テキストの影の色はこの値が初期値となります)。指定できる値は次の通りです。

colorに指定できる値

- **色**
  色の書式に従って任意の文字色を指定します。
- **transparent**
  文字色を透明にします。

opacityプロパティ

opacityプロパティを使用すると背景も含んだ要素全体の透明度を指定することができます。次の値が指定でき、初期値は1(不透明)となっています。

opacityに指定できる値

- **実数**
  透明度を0.0(透明)から1.0(不透明)の範囲の実数で指定します。

以下は、opacityプロパティの使用例です。
半透明であることが分かりやすいように、body要素に背景画像を指定した状態で、h1要素に「opacity: 0.5;」を指定しています。文字色は白で背景色は黒にしていますが、文字色も背景色も半透明になっていることが確認できます。サンプルファイルを用意してありますので、opacityプロパティの値を書き換えて、透明度が変化することを確認してみてください。

HTML　sample/chapter-05/lecture-5-2/01.html

```
<body>
<h1>
この要素全体の透明度をCSSのopacityプロパティで指定しています。
</h1>
</body>
```

CHAPTER 5

CSS　sample/chapter-05/lecture-5-2/01-opacity.css

```css
body {
  background-image: url(images/photo.jpg);
}
h1 {
  color: white;
  background-color: black;
  opacity: 0.5;
}
```

h1要素に「opacity: 0.5;」を指定



opacityプロパティの使用例



opacityプロパティの値を0.2に変えた状態



opacityプロパティの値を0.7に変えた状態

LECTURE 5-3

テキスト関連のプロパティ

ここからテキストに対して指定できるCSSプロパティの説明をしていきます。プロパティで使用する値の説明と、実用に際して気をつけなければいけないことをまとめています。

▶▶ フォント関連プロパティ

続けて、テキストに関連するCSSのプロパティを紹介します。まずはテキスト部分のフォントを制御するプロパティ(フォントサイズ・行間・フォントの種類・太字・斜体)からです。

font-sizeプロパティ

Chapter 2では、**font-sizeプロパティ**でh1要素とp要素の**フォントサイズを指定**しました。そこでは数値に「px」という単位をつけてピクセル数でフォントサイズを指定しましたが、実際には次のようなさまざまな値が指定できます(初期値は「medium」で、ブラウザで設定している標準フォントのサイズとなります)。

font-sizeに指定できる値

- **単位つきの実数**
  フォントサイズを単位つきの実数(16pxなど)で指定します。

- **パーセンテージ**
  親要素のフォントサイズに対するパーセンテージで指定します(150%のように、実数の直後に半角の「%」をつけて指定します)。

- **xx-small, x-small, small, medium, large, x-large, xx-large**
  7種類のキーワードで指定できます。
  xx-smallがもっとも小さく、mediumは普通のサイズで初期値、xx-largeがもっとも大きなサイズとなります(実際に表示されるサイズはブラウザによって異なります)。

- **smaller, larger**
  親要素のフォントサイズに対して、一段階小さく(smaller)、または大きく(larger)します。

CSSで使用する単位について

さて、font-sizeには「単位つきの実数」が指定できると書かれていますが、CSSでよく使われる主な単位をここで紹介しておきましょう。

単位	説明
px	ピクセル
pt	ポイント
cm	センチメートル
mm	ミリメートル
em	その要素のfont-sizeプロパティの値を1とする単位

CSSで使用できる主な単位

Chapter 2ですでに使用しましたが、pxは画面の1ピクセルを1とする単位です。ptは、ワープロなどでフォントサイズを指定するときに使われる単位のポイント(1/72インチ)と同じです。cmとmmは主に印刷用に用意されています(画面表示用のCSSでは基本的に使用されません)。

さて、最後のem(エムと読みます)ですが、この単位はfont-sizeプロパティ以外に使用した場合と、font-sizeプロパティに使用した場合とで意味が変わってきます。font-sizeプロパティ以外に使用した場合、この単位はその要素に指定されているfont-sizeプロパティの値(指定されていなければ初期値mediumの大きさ)を1とする単位となります。font-sizeプロパティに使用した場合は、親要素のfont-sizeプロパティの値を1とする単位となります。たとえば、親要素のfont-sizeプロパティの値が12pxの場合、「font-size: 1.5em;」と指定するとフォントサイズは18ピクセル(親要素の1.5倍)になるということです。emは比較的よく使用される単位ですので、しっかりと覚えておきましょう。

line-heightプロパティ

次に紹介するのは、行間(行の高さ)を指定するプロパティのline-height(ラインハイトと読みます)です。これもChapter 2で使用したプロパティの1つです。次の値が指定できます(初期値は「normal」です)。

line-heightに指定できる値

- **実数**
  行間を単位をつけない実数(1.5など)で指定します。
  行間は、ここで指定した実数とフォントサイズを掛けた高さとなります。

- **単位つきの実数**
  行間を単位つきの実数(24pxなど)で指定します。

- **パーセンテージ**
  行間をフォントサイズに対するパーセンテージ(150%など)で指定します。

- **normal**
  ブラウザ側で妥当だと判断する行間に設定します(ブラウザによって表示結果は異なります)。

指定できる値の中に「単位をつけない実数」が含まれています。フォントサイズを基準にしてその「何倍」と指定したいのであれば、emを使うこともパーセンテージを使うこともできるのに、なぜわざわざこの値が用意されているのでしょうか？
実は、これらの指定方法のあいだには、大きな違いがあります。どの方法で「何倍」と指定してもline-heightプロパティを指定した要素自体にはなんら違いは現れないのですが、その内部に含まれている要素には大きな違いが生じるのです。

CSSのプロパティの中には、表示指定をその要素だけに適用するものと、その内部に含まれる要素にも適用するものとがあります。たとえば、font-sizeは内部の要素にも適用されますが、背景関連のプロパティは内部の要素には適用されません。line-heightは、内部の要素にも適用されるプロパティです。

CSSでは、指定された値をこのように内部の要素に適用する場合、指定された値をそのまま適用するのではなく、計算結果の値を適用することになっています。たとえば、フォントサイズが10ピクセルの要素に対して「line-height: 1.5em;」や「line-height: 150%;」を指定した場合、内部の要素に適用されるのは「1.5em」や「150%」ではなく、計算結果の「15px」なのです。
この場合でも、内部の要素のフォントサイズがすべて同じであれば、特に影響はありません。しかし、内部にフォントサイズが30ピクセルのテキストが入っていたりすると、行間15ピクセルには収まりきらずにはみ出して（ほかの行と重なって）しまうことになります。

line-heightに指定できる「単位をつけない実数」は、その問題を解決するために用意されたものです。
実は、「単位をつけない実数」を指定した場合に限り、計算結果の値ではなく、その「単位をつけない実数」をそのまま内部の要素にも適用することになっています。こうすることで、内部に30ピクセルのテキストが入っていても、行間は「15px」ではなく「45px」になり、テキストが行をはみ出すことはなくなるというわけです。このような理由から、line-heightには多くの場合「単位をつけない実数」が指定されます。

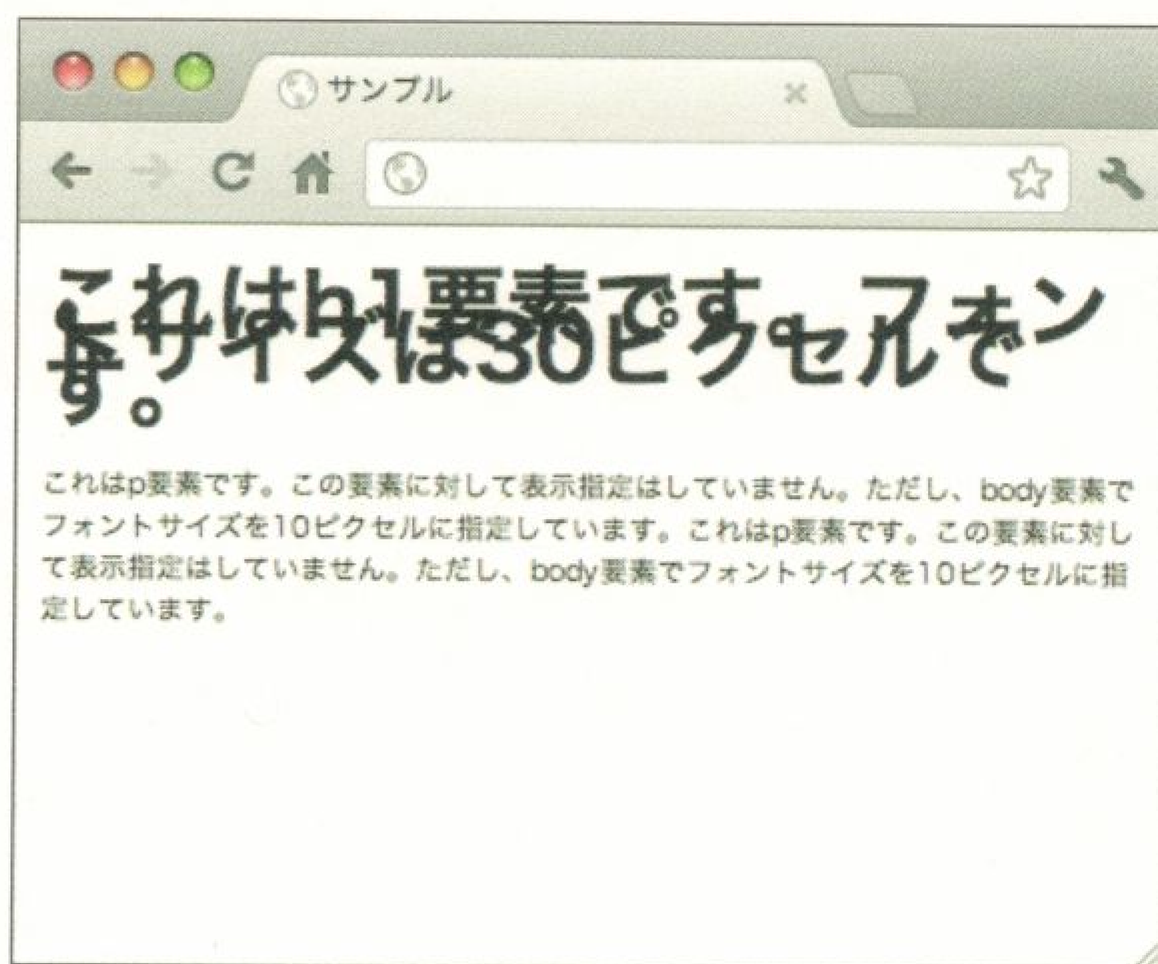
HTML　**sample/chapter-05/lecture-5-3/01.html**

```
<body>
<h1>
これはh1要素です。フォントサイズは30ピクセルです。
</h1>
<p>
これはp要素です。　・・・中略・・・　しています。
</p>
</body>
```

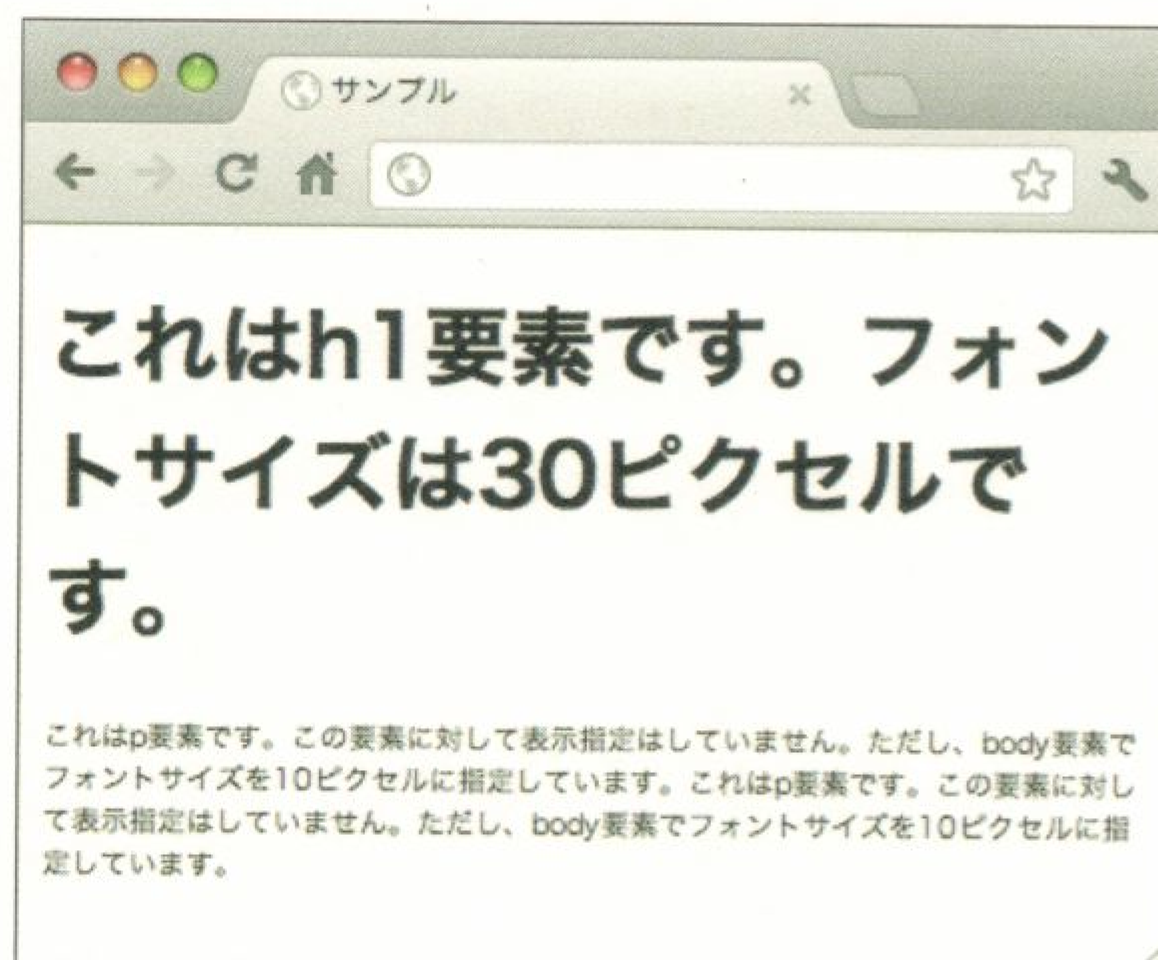
CSS　sample/chapter-05/lecture-5-3/01-line-height.css

```
body {
  font-size: 10px;
  line-height: 150%;
}
h1 {
  font-size: 30px;
}
```

body要素に「line-height: 150%;」を指定



上のソースコードをブラウザで表示させるとこのようになる



line-heightの値を「150%」から「1.5」へと変更すると文字が重ならなくなる

COLUMN

値の継承について

CSSのプロパティの中には、指定された値をその内部に含まれる要素にも適用するものとしないものがあると書きました。たとえば、body要素にfont-sizeプロパティを指定すると、その値はbody要素の中に含まれるp要素などにも適用されます。しかし、body要素にbackground-imageプロパティで背景画像を指定しても、内部のp要素などには適用されません。

このように、指定された値を内部の要素にも継承させるかどうかは、通常は特に意識しなくてもそのプロパティの種類に応じてうまく機能するように作られています。そして、親要素の値を継承しないプロパティであっても、値に「inherit」と指定するだけで強制的に親要素の値を継承させることが可能となっています。ただし、Internet Explorer 7以前のバージョンでは「inherit」に未対応ですので注意してください。

なお、「inherit」はCSSのすべてのプロパティで使用できますが、本書の「○○○に指定できる値」という部分では記載を省略しています。

font-familyプロパティ

さて、次もChapter 2で使用したプロパティの**font-family**です。テキストを表示させるフォントの種類を指定するためのプロパティで、値には次のようにフォントの名前またはフォントの種類をあらわすキーワードが指定できます。初期値はブラウザで設定されているフォントとなります。

font-familyに指定できる値

・フォント名

フォントの種類をフォントの名前で指定します。
ここで指定するフォント名は、「Arial Bold」や「Arial Italic」のような個別のフォント名ではなく、「Arial」のようなフォントファミリー名である点に注意してください(「Arial Bold」や「Arial Italic」を選択するには、次に説明するfont-weightプロパティとfont-styleプロパティを使用します)。

・serif, sans-serif, cursive, fantasy, monospace

フォントのおおまかな種類をあらわす5つのキーワードで指定できます(実際に表示されるフォントの種類はブラウザによって異なります)。

キーワード	フォントの種類
serif	明朝系フォント
sans-serif	ゴシック系フォント
cursive	草書体・筆記体系フォント
fantasy	ポップ系フォント
monospace	等幅フォント

ただし、閲覧者の環境に指定したフォントがインストールされていない場合に備えて、カンマ(,)で区切って複数の値を指定しておくことができます。その場合は、より前に(左側に)指定されているフォントで、その環境で利用可能なものが採用されることになります。

フォント名の中には、半角スペースや数字、記号などを含むものがあります。そのような名前を指定する場合には、名前の前後に引用符("または')をつけて名前が間違いなく認識されるようにすることが推奨されています(引用符をつけたりつけなかったりするよりは、常につけることに決めておいた方がより安全です)。ただし、フォントの種類をあらわす5つのキーワードに関しては、引用符をつけると認識されなくなりますので注意してください。

sample/chapter-05/lecture-5-3/02-font-family.css

```
body {
  font-family: "メイリオ", "Meiryo", "ヒラギノ角ゴ Pro W3", "Hiragino Kaku Gothic Pro", "ＭＳ Ｐゴシック", "MS P Gothic", "Osaka", sans-serif;
}
```

font-familyプロパティの使用例。同じフォントファミリーを日本語と英語の両方で指定しているのは古いブラウザにも認識させるため

CHAPTER 5

さて、font-familyプロパティの属性の解説部分でも触れましたが、1つのフォントファミリーの中にあるboldの書体を選択するのがfont-weightプロパティで、italicの書体を選択するのがfont-styleプロパティです。もし、boldやitalicの専用書体が用意されていないフォントであっても、一般的なワープロなどのように太字や斜体で表示させることが可能です。では、これらの使い方を順に見ていきましょう。

font-weightプロパティ

font-weightプロパティには次の属性が指定できます。初期値は「normal」です。

font-weightに指定できる値

・100, 200, 300, 400, 500, 600, 700, 800, 900
　9種類の数値で指定できます(実際に表示される太さはフォントの種類によって異なります)。100がもっとも細く、400が標準の太さで初期値、900が最も太くなります。

・bold
　そのフォントの一般的な太字の太さにします。700を指定した場合と同様の結果となります。

・normal
　そのフォントの標準の太さにします。400を指定した場合と同様の結果となります。

・bolder
　一段階太くします。

・lighter
　一段階細くします。

指定できる値としては、太さを細かく指定できるようになっていますが、日本語の環境では太さの異なる日本語フォントがそれほど多くインストールされていることは期待できません。そのため、値としてはシンプルに「bold」を使用するのが一般的です。strong要素のように最初から太字で表示されるフォントを標準状態に戻したい場合は、「normal」を指定します。

font-styleプロパティ

font-styleプロパティには次の属性が指定できます。初期値は「normal」です。

font-styleに指定できる値

・italic
　イタリック体(イタリック体専用にデザインされたフォント)で表示します。イタリック体がない場合は、「oblique」が指定されたものとして処理されます。

・oblique
　斜体(元の書体をシンプルに斜めにしたフォント)で表示します。斜体がない場合は、標準のフ

ォントを斜めに変換して表示します。

・normal
イタリック体や斜体ではない標準のフォントで表示します。

font-weightと同様に、一般的なユーザーの環境において、日本語フォントにイタリック体や斜体が用意されていることは期待できません。そのため、日本語を斜体で表示させたい場合は、値として「italic」を指定します(そうすると、イタリック体があればそれを採用し、それがなければ斜体を採用し、それもなければ標準のフォントを斜めに変換して表示されます)。em要素やi要素のように最初から斜体で表示されるフォントを標準状態に戻したい場合は、「normal」を指定します。

フォント関連の値をまとめて指定する

fontプロパティ

実は、ここまでに説明してきたフォント関連のプロパティの値を、まとめて一度に指定できるプロパティがあります。それが**fontプロパティ**です。
今のところCSS3の仕様ではこれ以外の値も指定できることになっていますが、日本語環境ではあまり使用されないもの(現時点ではほとんど使用できないもの)も含まれているため、本書では以下の値に限って指定方法を説明します。

fontに指定できる値

・**font-styleの値**
font-styleプロパティに指定できる値が指定できます。

・**font-weightの値**
font-weightプロパティに指定できる値が指定できます。

・**font-sizeの値**　※必須
font-sizeプロパティに指定できる値が指定できます。

・**line-heightの値**
line-heightプロパティに指定できる値が指定できます。

・**font-familyの値**　※必須
font-familyプロパティに指定できる値が指定できます。

値の指定順序について

line-heightの値を除き、各プロパティの値は半角スペースで区切って指定します。ただし、必要な値だけを任意の順序で指定できるわけではない点に注意してください。値は次のルールで指定します。

1. はじめに、必要に応じてfont-styleとfont-weightの値を指定します。順序はどちらが先でもかまいません。指定しなければ値はそれぞれnormalにリセットされます。
2. 次に、font-sizeの値を指定します。この値は省略できません。
3. 次に、必要に応じてline-heightの値を指定します。この値を指定する場合は、font-sizeの値とのあいだを半角のスラッシュ(/)で区切ります。指定しなければ値はnormalにリセットされます。
4. 最後に、font-familyの値を指定します。この値は省略できません。

sample/chapter-05/lecture-5-3/03-font.css

```
h1 {
  font: 24pt serif;
}
p {
  font: bold 13px/1.7 "メイリオ", "Meiryo", sans-serif;
}
```

fontプロパティの使用例

テキスト関連プロパティ

さて、続けてフォント以外のテキスト関連プロパティを紹介していきます。フォント関連プロパティは見た目の変化がちょっと地味でしたが、ここからは見た目もそれなりに変化します。

text-shadowプロパティ 【CSS3新】

はじめはChapter 2で使用した**text-shadowプロパティ**です。その名のとおり、テキストに影を表示させるプロパティです。

text-shadowに指定できる値

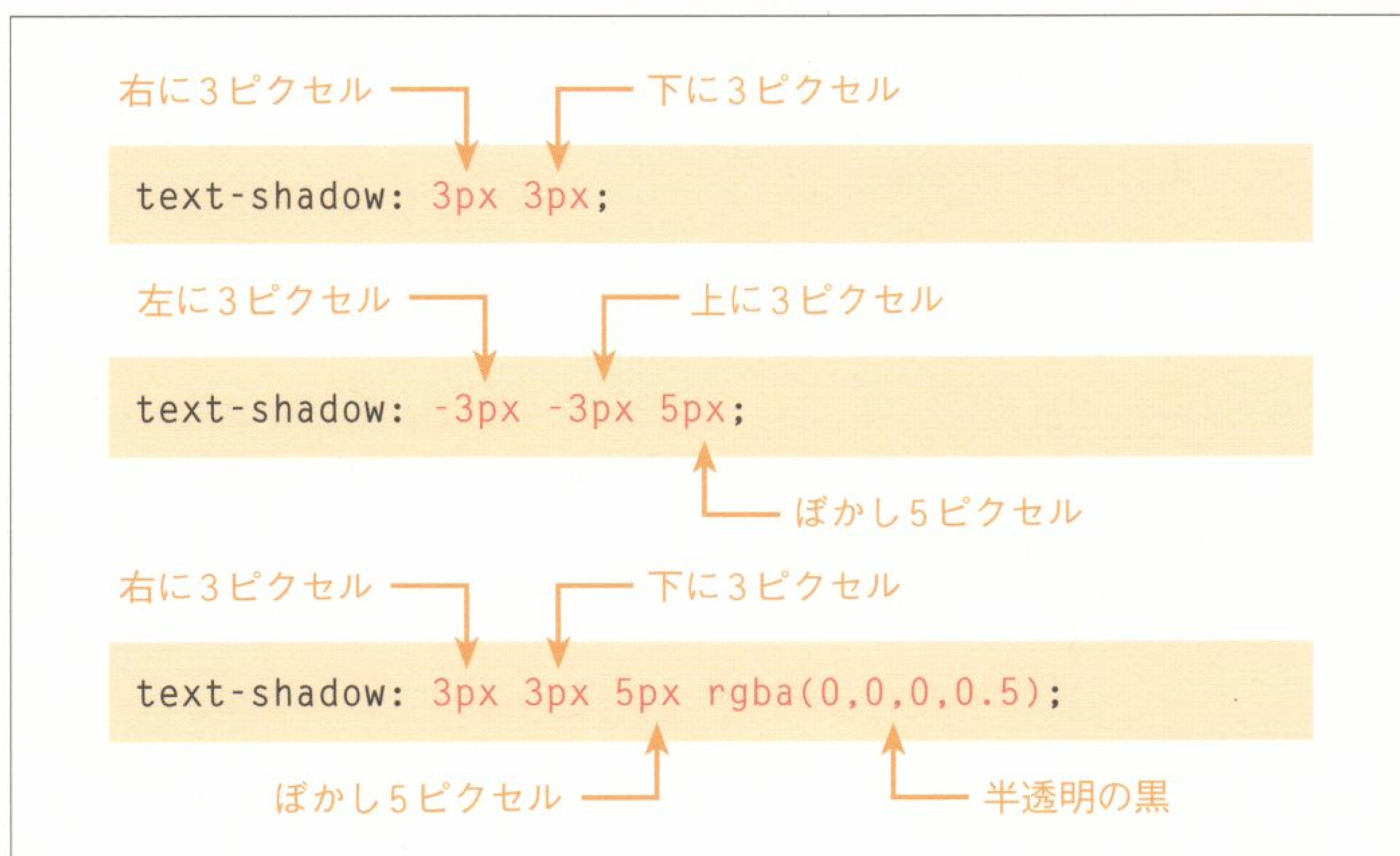
- **単位つきの実数**
  「影の表示位置」と「ぼかす範囲」は単位つきの実数で指定します。
- **色**
  色の書式に従って影の色を指定できます。
- **none**
  影を表示させません。この値は単独で指定します。

影の表示位置の指定方法

テキストに影を表示させる場合は、まず「単位つきの実数」で影の表示位置を指定します。影の表示位置は、元のテキストの位置を基準に、そこから右方向への移動距離、下方向への移動距離の順に半角スペースで区切って指定します。影を左方向、上方向に移動させたい場合は、それぞれマイナスの値を指定してください。

さらに半角スペースで区切って、影をぼかす範囲（ぼかしの強さ）を指定することもできます。

影の色を指定する場合は、これらの数値全体の前か後ろに、半角スペースで区切って指定します。色を指定しなければ、影はcolorプロパティの色で表示されます。



影のさまざまな指定方法の例

複数の影を指定する場合

また、影は1つだけでなく複数表示させることもできます。複数の影を表示させるには、通常の影の指定をカンマで区切って複数書くだけでOKです。その場合は、より前に（左側に）指定されている影の方が上に重なって表示されます。

HTML　sample/chapter-05/lecture-5-3/04.html

```
<h1>これはh1要素です。</h1>

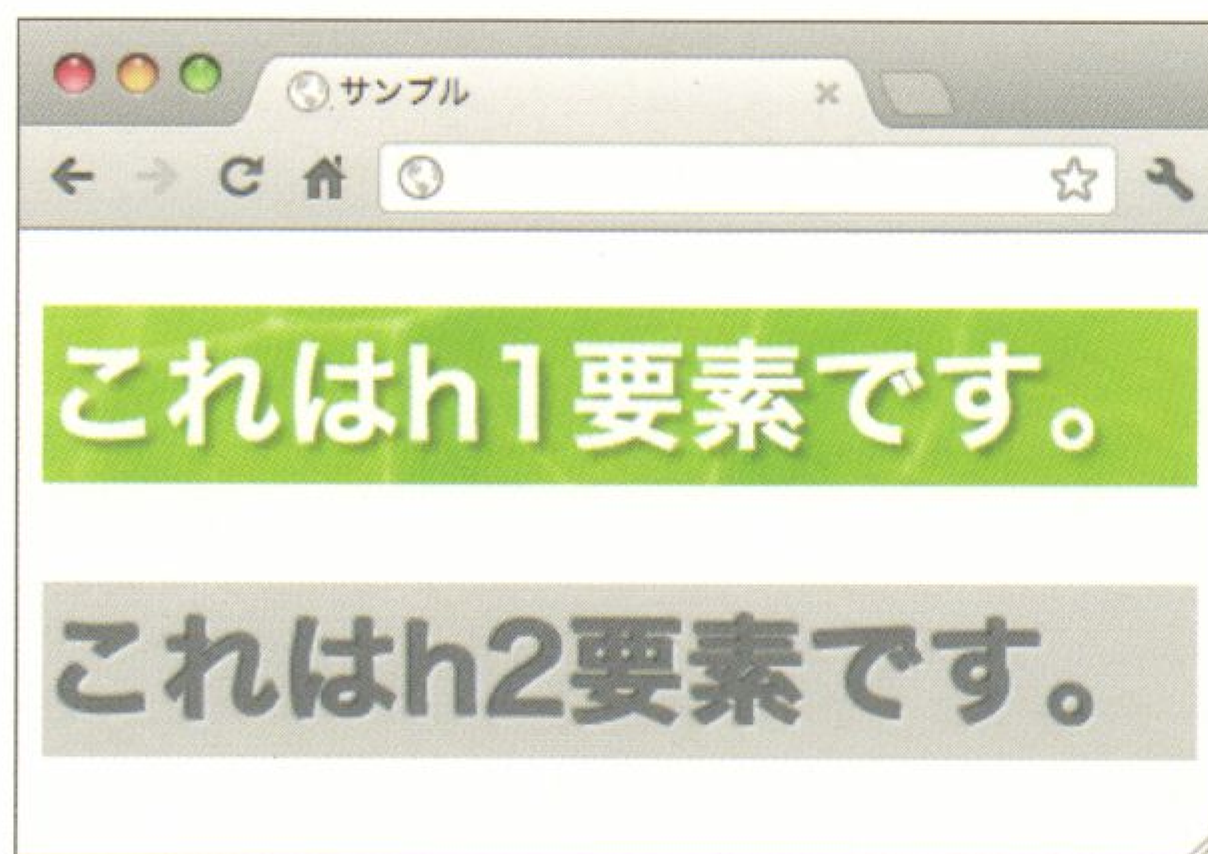
<h2>これはh2要素です。</h2>
```

CSS　sample/chapter-05/lecture-5-3/04-text-shadow.css

```
h1, h2 {
  font-size: 40px;
```

```
}
h1 {
  color: #fff;
  background-image: url(images/leaf.jpg);
  text-shadow: 3px 3px 5px rgba(0,0,0,0.5);
}
h2 {
  color: #666;
  background: #ccc;
  text-shadow: -1px -1px 2px #000, 1px 1px 2px #fff;
}
```

text-shadowプロパティの使用例。h1要素の影には半透明の黒を指定。h2要素は、左上に黒い影、右下に白い影を表示させている



上のソースコード例を表示させたところ

text-alignプロパティ

次は、行揃えを指定する**text-alignプロパティ**です。

このプロパティは、ブロックレベル要素にしか指定できない点に注意してください(ブロックレベル要素に指定して、その内容であるインライン要素全体の行揃えを設定します)。次の値が指定できます。

text-alignに指定できる値

- `left`
  インライン要素は、左揃えで表示されます。
- `right`
  インライン要素は、右揃えで表示されます。
- `center`
  インライン要素は、中央揃えで表示されます。

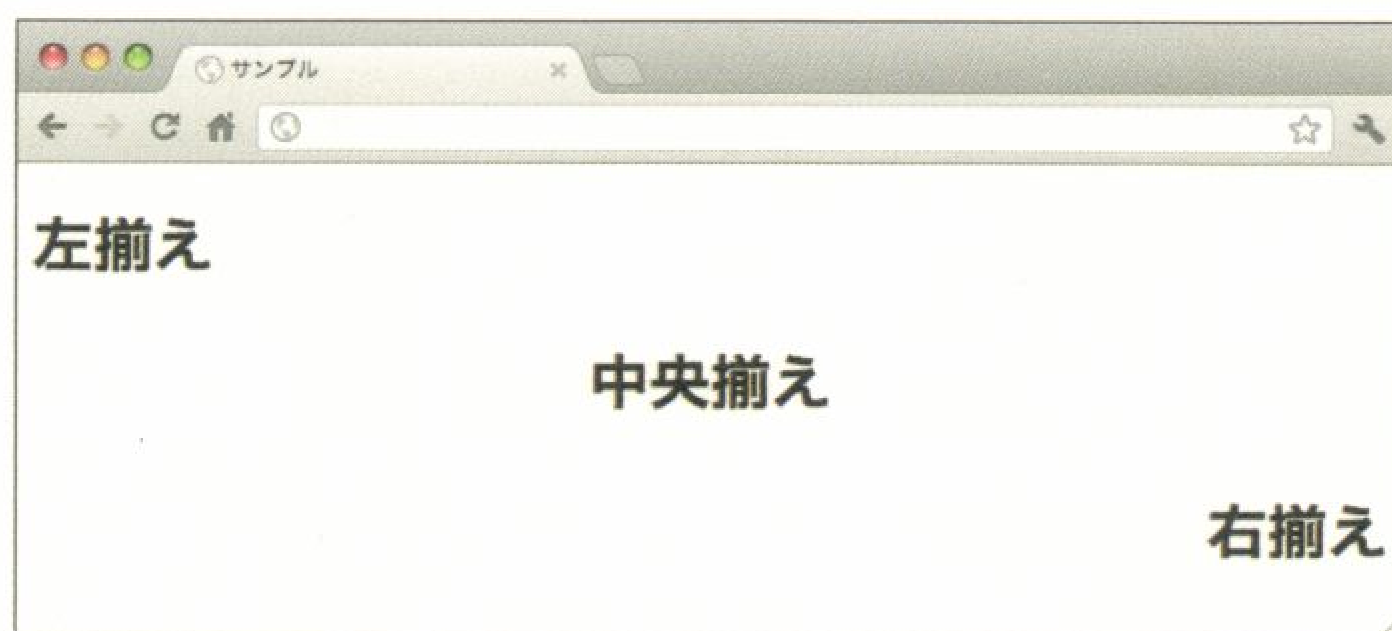
HTML　sample/chapter-05/lecture-5-3/05.html

```
<h1>左揃え</h1>
<h2>中央揃え</h2>
<h3>右揃え</h3>
```

CSS　sample/chapter-05/lecture-5-3/05-text-align.css

```
h1, h2, h3 {
  font-size: 30px;
}
h1 { text-align: left; }
h2 { text-align: center; }
h3 { text-align: right; }
```

text-align プロパティの使用例



上のソースコード例を表示させたところ

text-decoration プロパティ

次は、主に**テキストに下線**を表示させる際に使用する**text-decoration プロパティ**です。次の値が指定できます。リンクのようにはじめから下線が表示されているテキストの下線を消すには「none」を指定します。

text-decorationに指定できる値

- **underline**
  下線を表示させます。
- **overline**
  上線を表示させます。
- **line-through**
  取消線を表示させます。
- **none**
  テキストの線を消します。

CHAPTER 5

HTML　sample/chapter-05/lecture-5-3/06.html

```html
<h1>下線</h1>
<h2>上線</h2>
<h3>取消線</h3>
```

CSS　sample/chapter-05/lecture-5-3/06-text-decoration.css

```css
h1, h2, h3 {
  font-size: 24px;
  font-weight: normal;
}
h1 { text-decoration: underline; }
h2 { text-decoration: overline; }
h3 { text-decoration: line-through; }
```

text-decorationプロパティの使用例



上のソースコード例を表示させたところ

letter-spacingプロパティ

次は**文字間隔**を指定する**letter-spacingプロパティ**です。標準の文字間隔の状態から、どれだけ間隔を開くのかを指定します。マイナスの値を指定して文字間隔を狭くすることもできます。次の値が指定できます。

letter-spacingに指定できる値

- **単位つきの実数**
  文字間隔を単位つきの実数で指定します。

- `normal`
  文字間隔を標準の状態にします。

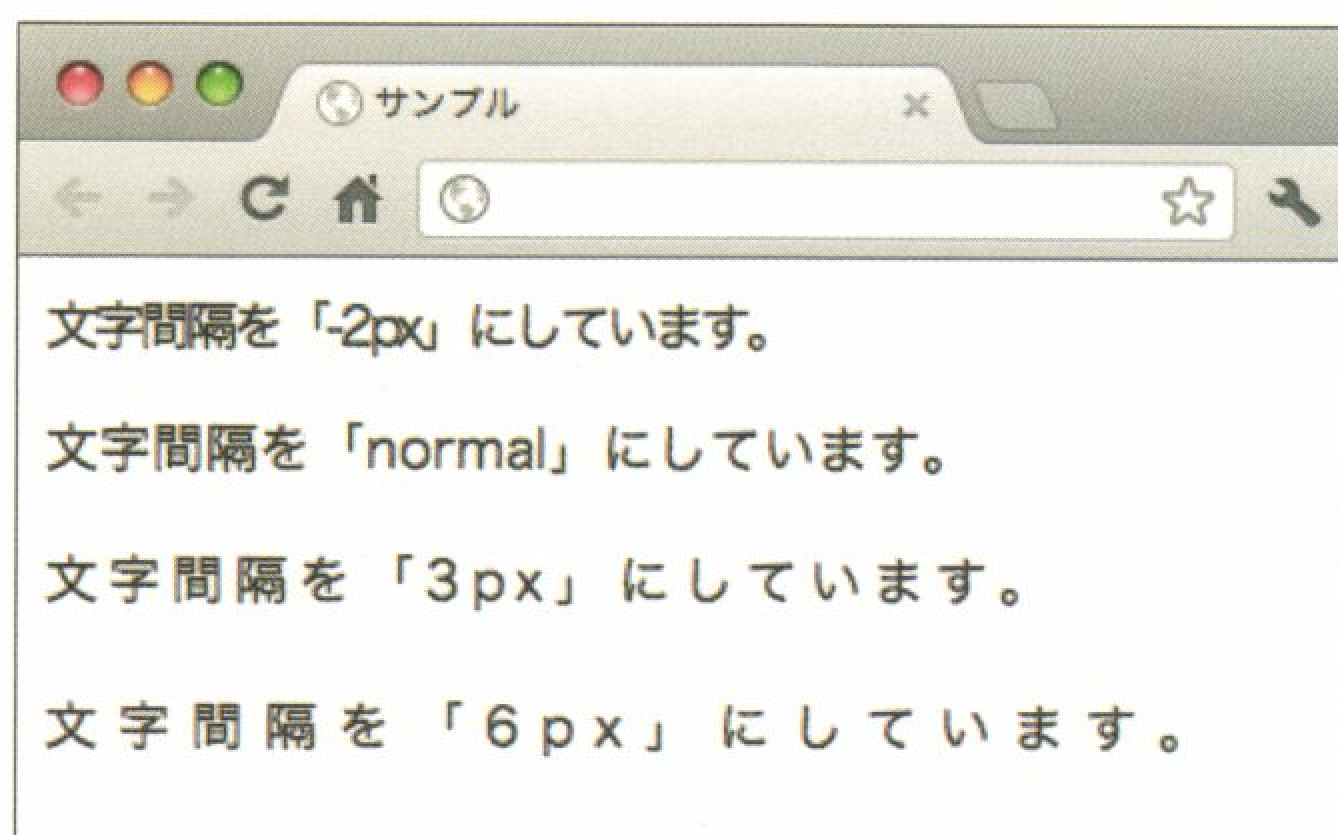
HTML　sample/chapter-05/lecture-5-3/07.html

```
<h1>文字間隔を「-2px」にしています。</h1>
<h2>文字間隔を「normal」にしています。</h2>
<h3>文字間隔を「3px」にしています。</h3>
<h4>文字間隔を「6px」にしています。</h4>
```

CSS　sample/chapter-05/lecture-5-3/07-letter-spacing.css

```
h1, h2, h3, h4 {
  font-size: 16px;
  font-weight: normal;
}
h1 { letter-spacing: -2px; }
h2 { letter-spacing: normal; }
h3 { letter-spacing: 3px; }
h4 { letter-spacing: 6px; }
```

letter-spacingプロパティの使用例



上のソースコード例を表示させたところ

text-indentプロパティ

次はブロックレベル要素の**1行目のインデント**（字下げ）の量を指定する**text-indentプロパティ**です。通常はp要素に適用しますが、それ以外のブロックレベル要素にも適用できます。初期値は0です。p要素に対して「1em」を指定すると、段落の先頭がほぼ1文字分あきます。次の値が指定できます。

text-indentに指定できる値

- **単位つきの実数**
  インデントの量を単位つきの実数で指定します。
- **パーセンテージ**
  インデントの量を幅に対するパーセンテージで指定します。

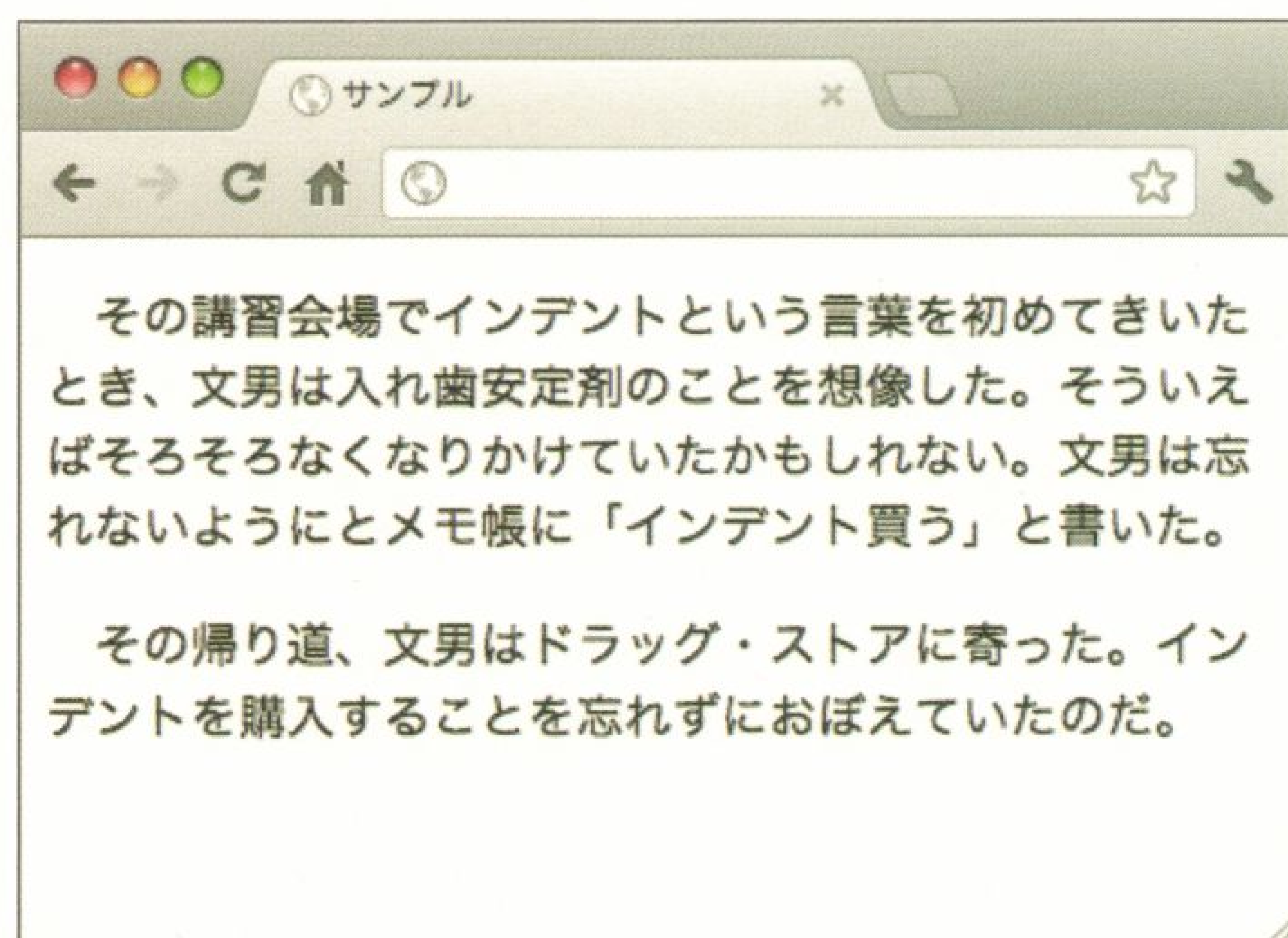
HTML　sample/chapter-05/lecture-5-3/08.html

```
<p>
その講習会場でインデントという言葉を初めてきいたとき、文男は入れ歯安定剤のことを想像した。そういえばそろそろなくなりかけていたかもしれない。文男は忘れないようにとメモ帳に「インデント買う」と書いた。
</p>
<p>
その帰り道、文男はドラッグ・ストアに寄った。インデントを購入することを忘れずにおぼえていたのだ。
</p>
```

CSS　sample/chapter-05/lecture-5-3/08-text-indent.css

```
p { text-indent: 1em; }
```

text-indentプロパティの使用例



上のソースコード例を表示させたところ

text-transformプロパティ

次の**text-transformプロパティ**を使用すると、アルファベットを**大文字**で表示させたり、**小文字**で表示させたりすることができます。
もちろん、ソースコード上ですべて大文字で書けばそのまま大文字で表示されることになりますが、表示上大文字にするのか小文字にするのかといった制御に関しては、Webページの元データであるHTMLのソースコード自体を書き換えずに、**CSS側で対処する**方がスマートです。しかも、アルファベットをすべて大文字で入力してしまうと、音声で読み上げさせるときに正しく読み上げられなくなる可能性もあります(環境やその単語にもよりますが、すべて大文字にすると単語として発音されずに、1文字ずつアルファベットで読み上げられる場合もあります)。
次の値が指定できます。

text-transformに指定できる値

- **uppercase**
  半角のアルファベットをすべて大文字で表示させます。

- **lowercase**
  半角のアルファベットをすべて小文字で表示させます。

- **capitalize**
  半角のアルファベットのうち、各単語の先頭の文字だけを大文字で表示させます。

- **none**
  テキストを元の状態のまま表示させます。

HTML　sample/chapter-05/lecture-5-3/09.html

```html
<h1>This property transforms text.</h1>
<h2>This property transforms text.</h2>
<h3>This property transforms text.</h3>
<h4>This property transforms text.</h4>
```

CSS　sample/chapter-05/lecture-5-3/09-text-transform.css

```css
h1, h2, h3, h4 {
  font-size: 16px;
  font-weight: normal;
}
h1 { text-transform: uppercase; }
h2 { text-transform: lowercase; }
h3 { text-transform: capitalize; }
h4 { text-transform: none; }
```

text-transformプロパティの使用例



上のソースコード例を表示させたところ

white-spaceプロパティ

通常、HTMLでは半角スペース・改行・タブは(連続していればそれらをまとめて)1つの半角スペースに変換して表示しますが、**white-spaceプロパティ**を使用するとその表示方法を変更することができます。指定する値によって、それぞれ次のような表示となります。

white-spaceに指定できる値

- `normal`
  半角スペース・改行・タブは、(連続していればまとめて)半角スペースに変換して表示します。幅がいっぱいになると自動的に行を折り返します。

- `nowrap`
  半角スペース・改行・タブは、(連続していればまとめて)半角スペースに変換して表示します。自動的な行の折り返しはしせん。

- `pre`
  半角スペース・改行・タブは、そのまま入力されている通りに表示します。自動的な行の折り返しはしせん。

- `pre-wrap`
  半角スペース・改行・タブは、そのまま入力されている通りに表示します。幅がいっぱいになると自動的に行を折り返します。

- `pre-line`
  半角スペースとタブは、(連続していればまとめて)半角スペースに変換して表示します。改行については、そのまま入力されている通りに表示します。幅がいっぱいになると自動的に行を折り返します。

ただし、値「`pre-wrap`」と「`pre-line`」に関しては、2011年に正式な仕様として公開されたCSS2.1で新しく追加されたもので、Internet Explorer 8以前は未対応である点に注意してください。

▶▶ ベンダープレフィックスとは?

テキスト関連プロパティの最後に、縦書き用のプロパティを2つ紹介します。しかし、それらを使用するには**ベンダープレフィックス**というものを知っておく必要がありますのでここで説明しておきましょう。

ここまでに登場したプロパティは、基本的にはCSS2.1以前の仕様に含まれていた(つまりすでに最終的に仕様が確定している)ものですので、とくにベンダープレフィックスなどは気にすることもなく使用できました。しかしご存知の通り、CSS3から新しく追加された機能の多くはまだ策定中で、その仕様は確定していません。つまり、現在のCSS3の仕様はすでに何度か変更されている

可能性があり、これからも変更される可能性があるということです。そのような機能を各社ブラウザがそのまま実装してしまうと、同じ機能でもブラウザの種類やバージョンによって表示結果が異なってしまうという事態が発生します。

そこで、ブラウザが草案（ワーキング・ドラフト）段階の機能を実装する際は、プロパティ名や関数名などの直前にベンダープレフィックスと呼ばれる接頭辞をつけて、正式名称とは異なる名前に変えて実装することになっています。ベンダープレフィックスとは、ベンダー（ブラウザやそのエンジンを開発している企業・組織）ごとに固有のプレフィックス（接頭辞）のことで、一般的なブラウザでは次のものが使用されています。

ブラウザ	ベンダープレフィックス
Google Chrome	-webkit-
Safari	-webkit-
Firefox	-moz-
Internet Explorer	-ms-
Opera	-o-

ブラウザと使用されているベンダープレフィックス

草案段階から一歩進んで勧告候補となった仕様に含まれている機能については、ベンダープレフィックスは基本的にはつけられません。“基本的には”と書いたのは、いったん勧告候補となった仕様が、草案に逆戻りしてしまうケースなどもあるからです。このように、ベンダープレフィックスが必要かどうかの境界は意外と曖昧です。そのようなわけで、基本的にはベンダープレフィックスをつけておき、そのあとにベンダープレフィックスをつけない指定も併記しておくと良いでしょう。

同じブラウザでもベンダープレフィックスつきの名前とついていない名前の両方をサポートしていることがあります。その場合はベンダープレフィックス無しの方がより新しい仕様に沿った実装をしている可能性が高いので、ベンダープレフィックスをつけない指定を必ず最後に指定しておくようにしてください。そうすることで、古い仕様に沿った実装は新しい実装に上書きされて、より正しい表示結果が得られるようになります。

▶▶ 縦書き用プロパティ

writing-modeプロパティ 【CSS3新】

ではさっそくベンダープレフィックスが必要となるCSS3の**writing-modeプロパティ**を使用してみましょう。このプロパティは、横書き（horizontal）か縦書き（vertical）かを指定するプロパティです。ただし、縦書きでも右から左へ読む場合と、左から右へと読む場合が考えられますので、その方向も含めた値が用意されています。指定できる値は次の通りです。

writing-modeに指定できる値

- **horizontal-tb**
  横書きにします。tbは上から下(top-to-bottom)の意味です。

- **vertical-rl**
  右から左へと進む縦書きにします。rlは右から左(right-to-left)の意味です。

- **vertical-lr**
  左から右へと進む縦書きにします。lrは左から右(left-to-right)の意味です。

ただし、2012年7月の時点でこのプロパティに対応しているのは、Google ChromeとSafariだけです。しかし、近い将来にInternet ExplorerやOperaも対応する可能性があるため、その備えとしてそれらのベンダープレフィックスつきの指定も追加してあります。また、Windows環境の場合は縦書きにするだけでなく、縦書き用のフォント(先頭が@ではじまるフォント)を指定する必要がある点に注意してください(そうしなければ各文字が90度回転した状態で表示されます)。

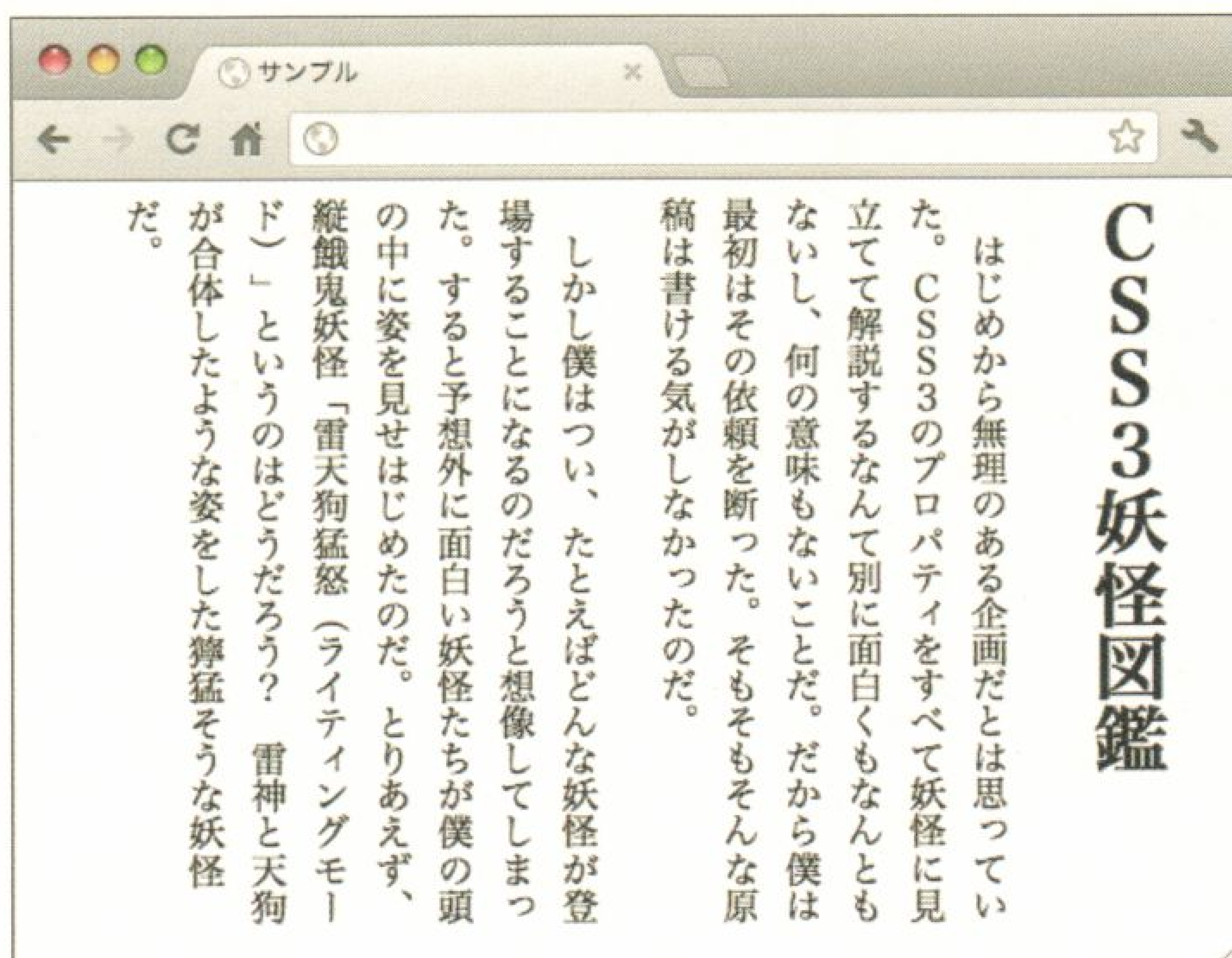
HTML　　sample/chapter-05/lecture-5-3/10.html

```
<body>
<h1>ＣＳＳ３妖怪図鑑</h1>
<p>
　はじめから無理のある企画だとは思っていた。ＣＳＳ３のプロパティをすべて妖怪に見立てて解説するなんて別に面白くもなんともないし、何の意味もないことだ。だから僕は最初はその依頼を断った。そもそもそんな原稿は書ける気がしなかったのだ。
</p>
<p>
　しかし僕はつい、たとえばどんな妖怪が登場することになるのだろうと想像してしまった。すると予想外に面白い妖怪たちが僕の頭の中に姿を見せはじめたのだ。とりあえず、縦餓鬼妖怪「雷天狗猛怒(ライティングモード)」というのはどうだろう？　雷神と天狗が合体したような姿をした獰猛そうな妖怪だ。
</p>
<body>
```

CSS　　sample/chapter-05/lecture-5-3/10-writing-mode.css

```
body {
  -webkit-writing-mode: vertical-rl;
  -moz-writing-mode: vertical-rl;
  -ms-writing-mode: vertical-rl;
  -o-writing-mode: vertical-rl;
  writing-mode: vertical-rl;
  font-family: "@ＭＳ 明朝", serif;
  line-height: 1.7;
}
```

writing-modeプロパティの使用例。Windows環境でも正しく表示させるには、先頭が@ではじまる縦書き用のフォント(この例では「@ＭＳ 明朝」)を指定する必要がある



前ページのソースコード例を表示させたところ

text-emphasis-style プロパティ 【CSS3 新】

続けて、圏点を表示させるために使用する **text-emphasis-style プロパティ** も紹介しておきましょう。圏点とは、縦書きの小説などで特定の部分を強調する目的でテキストの右側につけられている点のことです。次の値が指定できます。

text-emphasis-style に指定できる値

- `none`
  圏点を表示させません。
- `filled`
  塗りつぶした圏点にします。
- `open`
  塗りつぶさない圏点にします。
- `sesame`
  一般的なゴマ型の圏点を表示させます。
- `dot`
  小さな丸の圏点を表示させます。
- `circle`
  大きな丸の圏点を表示させます。
- `double-circle`
  二重丸の圏点を表示させます。
- `triangle`
  三角の圏点を表示させます。
- 文字
  圏点として表示させる文字を指定します。

このプロパティに対応しているのも、2012年7月の時点ではGoogle ChromeとSafariだけですが、将来に備えてほかのブラウザのベンダープレフィックスをつけた指定も組み込んであります。

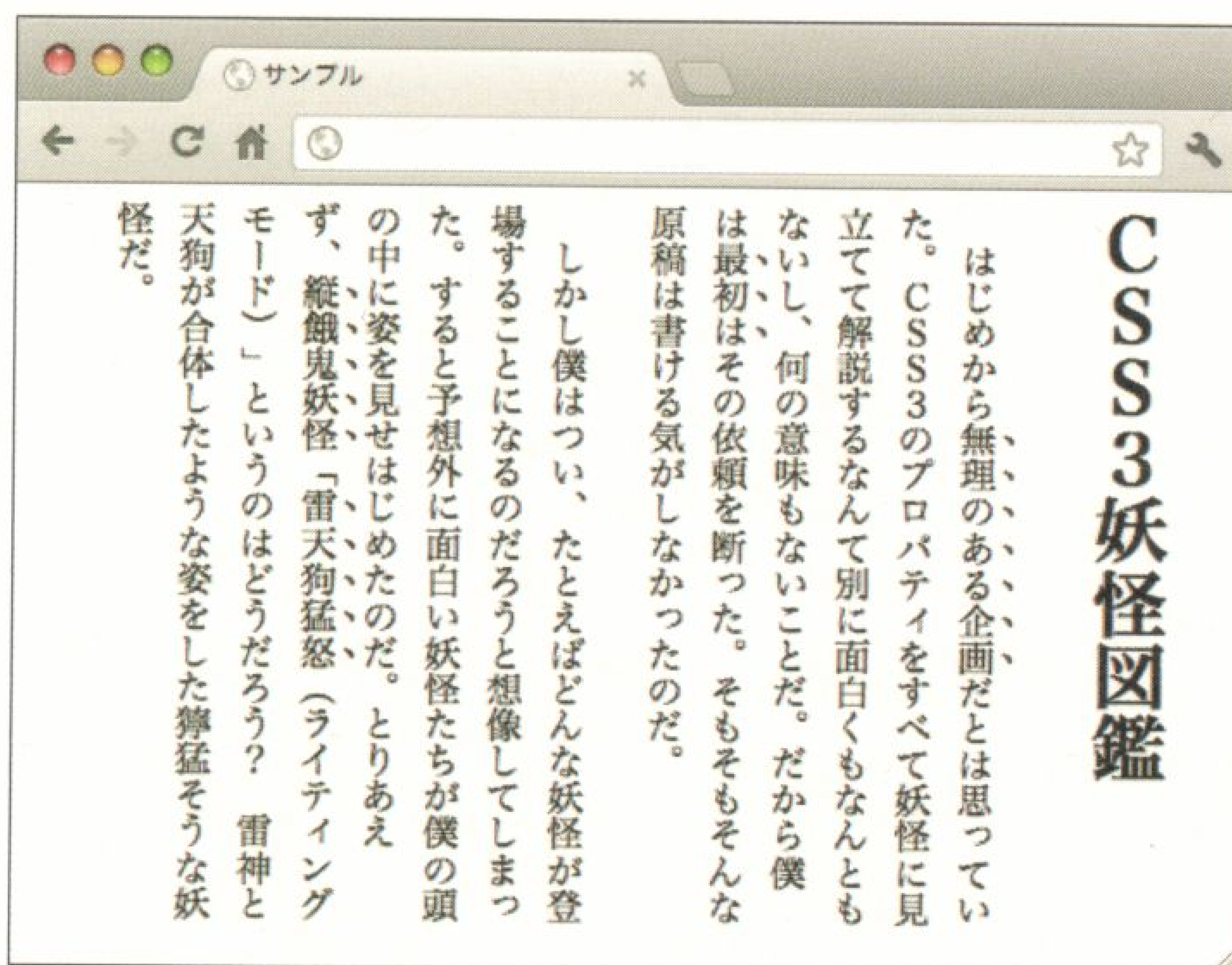
HTML　**sample/chapter-05/lecture-5-3/11.html**

```
<body>
<h1>ＣＳＳ３妖怪図鑑</h1>
<p>
　はじめから<em>無理のある企画</em>だとは思っていた。ＣＳＳ３のプロパティをすべて妖怪に見立てて解説するなんて別に面白くもなんともないし、何の意味もないことだ。だから僕は<em>最初は</em>その依頼を断った。そもそもそんな原稿は書ける気がしなかったのだ。
</p>
<p>
　しかし僕はつい、たとえばどんな妖怪が登場することになるのだろうと想像してしまった。すると予想外に面白い妖怪たちが僕の頭の中に姿を見せはじめたのだ。とりあえず、<em>縦餓鬼妖怪</em>「<em>雷天狗猛怒</em>（ライティングモード）」というのはどうだろう？　雷神と天狗が合体したような姿をした獰猛そうな妖怪だ。
</p>
</body>
```

CSS　**sample/chapter-05/lecture-5-3/11-text-emphasis.css**

```
em {
  font-style: normal;
  -webkit-text-emphasis-style: sesame;
  -moz-text-emphasis-style: sesame;
  -ms-text-emphasis-style: sesame;
  -o-text-emphasis-style: sesame;
  text-emphasis-style: sesame;
}
```

text-emphasis-styleプロパティの使用例。縦書きにする部分のソースコードは前のサンプルとまったく同じなので省略



上のソースコード例を表示させたところ

CHAPTER 6

CSSの適用先の指定方法

ここまでのサンプルでは、CSSの適用先として要素名だけを指定してきました。しかし、CSSがより実践的なものとなってくると、当然それだけでは対応しきれなくなってきます。Chapter 6では、CSSのセレクタの指定方法について一通りまとめて学習します。

LECTURE 6-1

よく使う主要なセレクタ

Lecture 3-4で一度説明していますが、CSSをどの要素に対して適用するかを指定するのが「セレクタ」です。ここまでは「要素名」をセレクタに使用してきましたが、セレクタには他にもいろいろな種類があります。

ブラウザの対応状況に注意

CSS3で使用できるセレクタは、約40種類あります※8。ただし、Internet Explorerはバージョン9でやっとそのすべてに対応したものの、それより前のバージョンでは多くのセレクタに未対応となっています。そのため、現時点でCSS3のセレクタを使用する際は、Internet Explorerの対応状況に注意する必要があります。

そのような状況のため、実際の制作で頻繁に使用されているセレクタは、全体の中のごく一部(CSS3以前から利用できたものの中の更に一部)です。そこで、まずはそのような一般的なブラウザのほとんどが対応している主要なセレクタから紹介しましょう。

なお、CSS3のセレクタに関する仕様は2011年9月29日の段階で正式に確定したものとして公開されています。そのため、セレクタに関してはベンダープレフィックス(p.102参照)をつける必要は一切ありません。

※8：ここで約40種類としているのは、すでに正式な仕様として公開されている「Selectors Level 3」で定義されているセレクタのことです。2012年7月段階ではまだ草案の「CSS Basic User Interface Module Level 3」には、それ以外のセレクタも含まれています。

▶▶ タイプセレクタ

ここまでのサンプルで使用してきたように、「要素名」をそのまま使って指定するセレクタのことをタイプセレクタと言います（要素のタイプ、つまり要素の種類で指定するセレクタという意味です）。

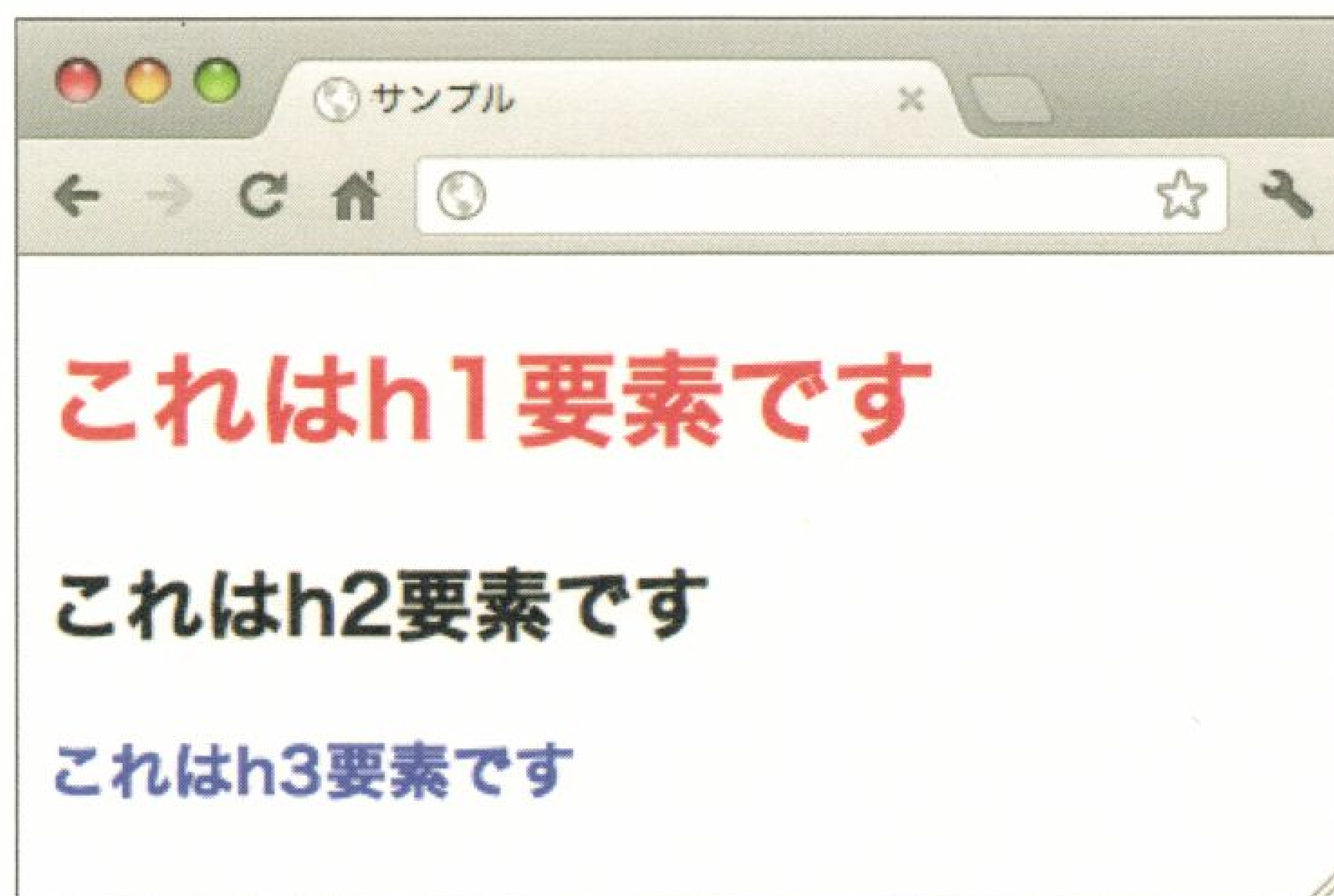
HTML　sample/chapter-06/lecture-6-1/01.html

```
<h1>これはh1要素です</h1>
<h2>これはh2要素です</h2>
<h3>これはh3要素です</h3>
```

CSS　sample/chapter-06/lecture-6-1/01-type.css

```
h1 { color: red; }
h2 { color: black; }
h3 { color: blue; }
```

タイプセレクタの使用例



上のソースコード例を表示させたところ

▶▶ユニバーサルセレクタ

「要素名」の代わりに「*」を指定すると、すべての要素に適用されます。このセレクタはユニバーサルセレクタと呼ばれています。

ただし、「*」の直後にほかのセレクタが続く場合に限り、「*」を省略して書くことができます(次の「クラスセレクタ」のところで詳しく説明します)。

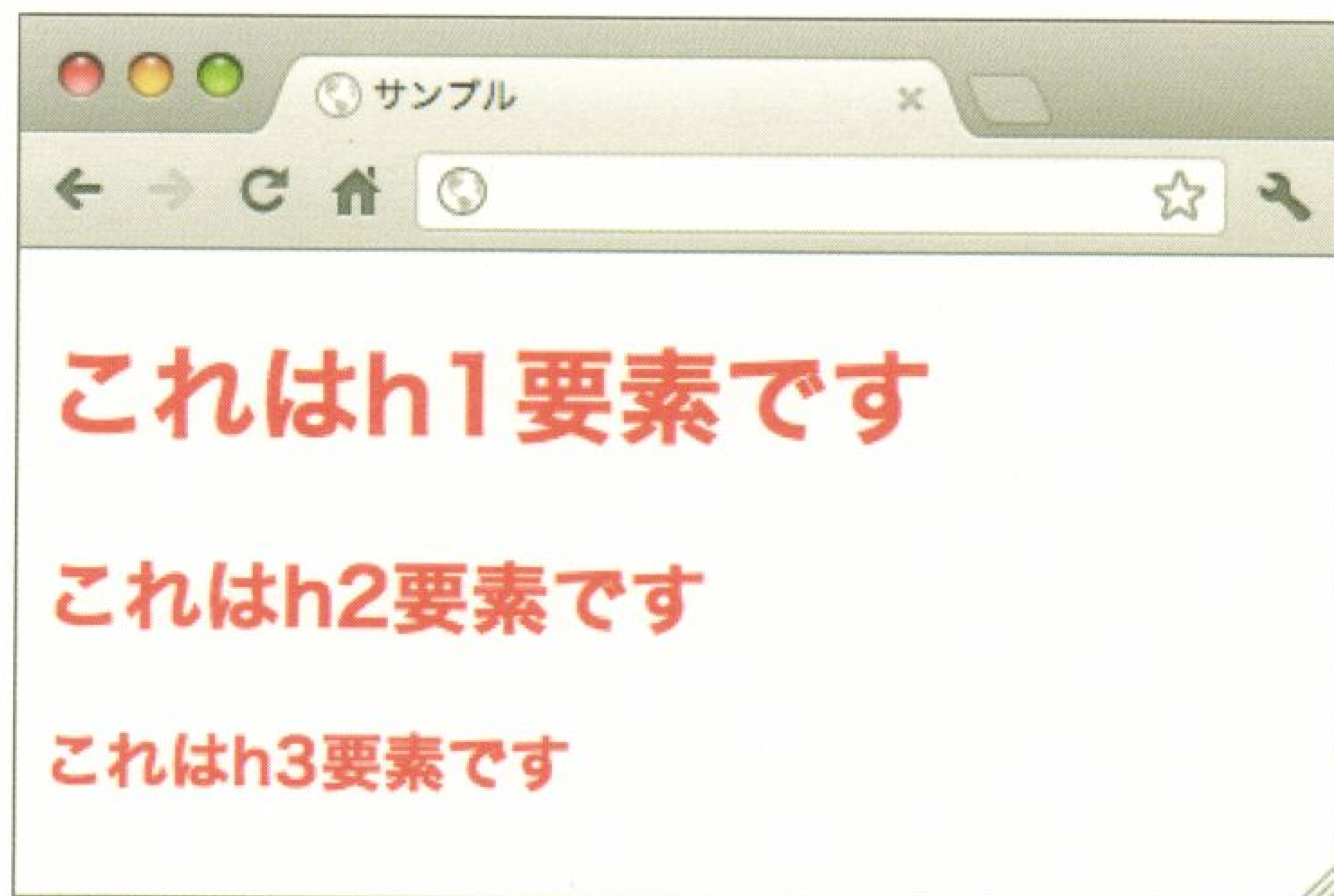
HTML　sample/chapter-06/lecture-6-1/02.html

```
<h1>これはh1要素です</h1>
<h2>これはh2要素です</h2>
<h3>これはh3要素です</h3>
```

CSS　sample/chapter-06/lecture-6-1/02-universal.css

```
* { color: red; }
```

ユニバーサルセレクタの使用例



上のソースコード例を表示させたところ

▶▶クラスセレクタ

class属性に特定の値が指定されている要素を適用対象とすることができます。このセレクタを使用する場合、はじめに「要素の名前」または「*」を書きます。これは、適用対象を特定の要素に限定する場合は「要素の名前」を指定し、要素を得に限定しない場合は「*」を使用するということです。あとは、その直後にピリオド(.)をつけ、さらに続けてclass属性の値を追加するだけでOKです。

このように「*」の直後に何かが続く場合には、「*」を省略することができます。たとえば、class属性の値として「sample」が指定されている要素を適用対象としたい場合には、次のように書くことができます。

```
p.sample {・・・} ←― class="sample" が指定されているp要素に適用
*.sample {・・・} ←― class="sample" が指定されている任意の要素に適用
.sample {・・・} ←― 「*」を省略した書き方
```

クラスセレクタの書き方のパターン

class属性の値は半角スペースで区切って複数指定できますが、この指定方法では指定された値が複数の値のうちのどれか1つと合致すればCSSが適用されることになります。

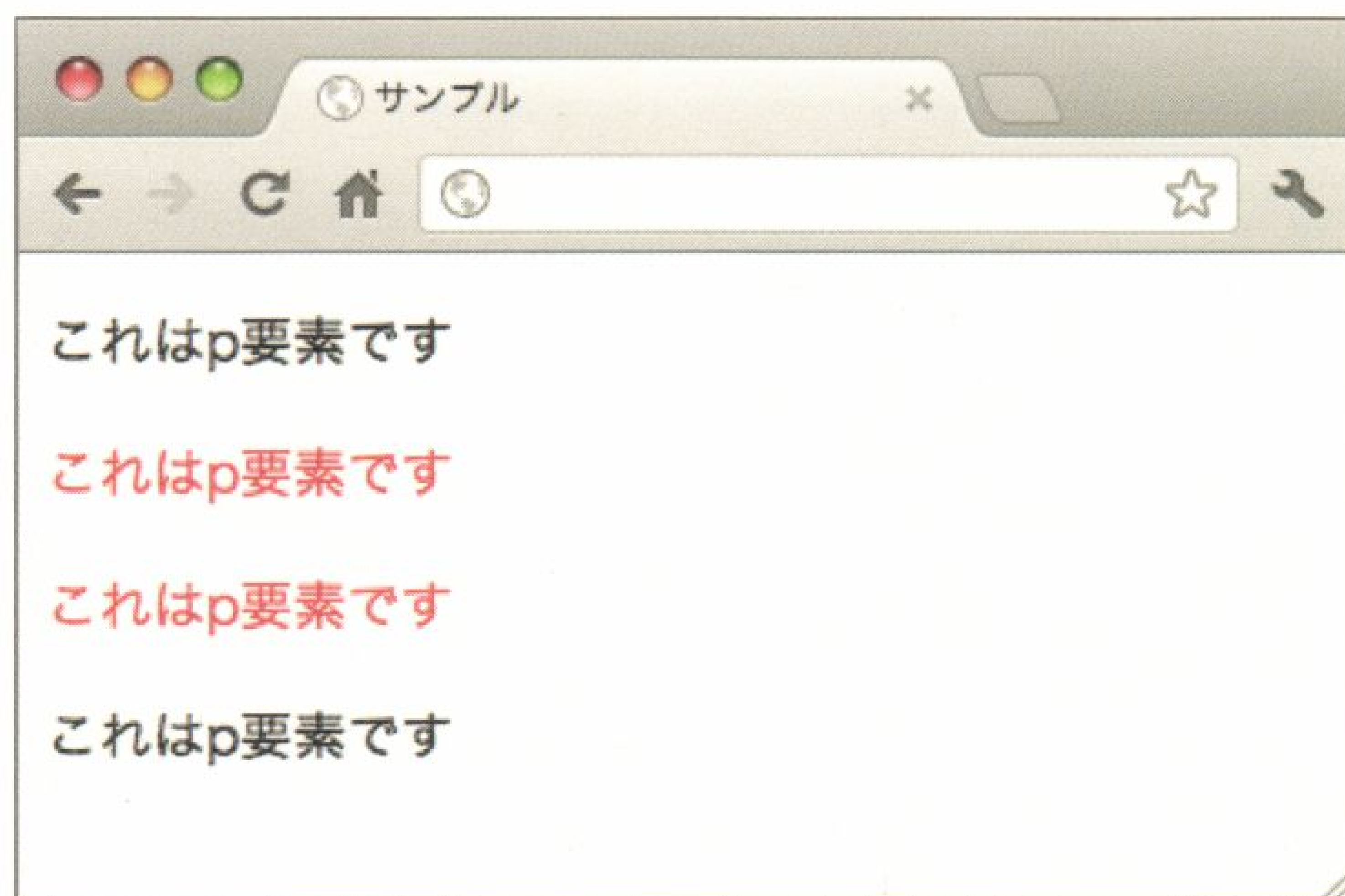
HTML　sample/chapter-06/lecture-6-1/03.html

```
<p>これはp要素です</p>
<p class="abc">これはp要素です</p>
<p class="abc def ghi">これはp要素です</p>
<p>これはp要素です</p>
```

CSS　sample/chapter-06/lecture-6-1/03-class.css

```
.abc { color: red; }
```

クラスセレクタの使用例



上のソースコード例を表示させたところ

▶▶IDセレクタ

クラスセレクタと同様に、id属性に特定の値が指定されている要素を適用対象とすることができます。書き方もクラスセレクタと同様ですが、IDセレクタの場合は「.」ではなく「#」を使います。この場合も、「*」は省略可能です。
なお、id属性はclass属性とは異なり、半角スペースで区切って複数の値を指定することはできない点に注意してください。

```
p#sample {・・・} ←── id="sample" が指定されているp要素に適用
*#sample {・・・} ←── id="sample" が指定されている任意の要素に適用
#sample {・・・} ←── 「*」を省略した書き方
```

クラスセレクタの書き方のパターン

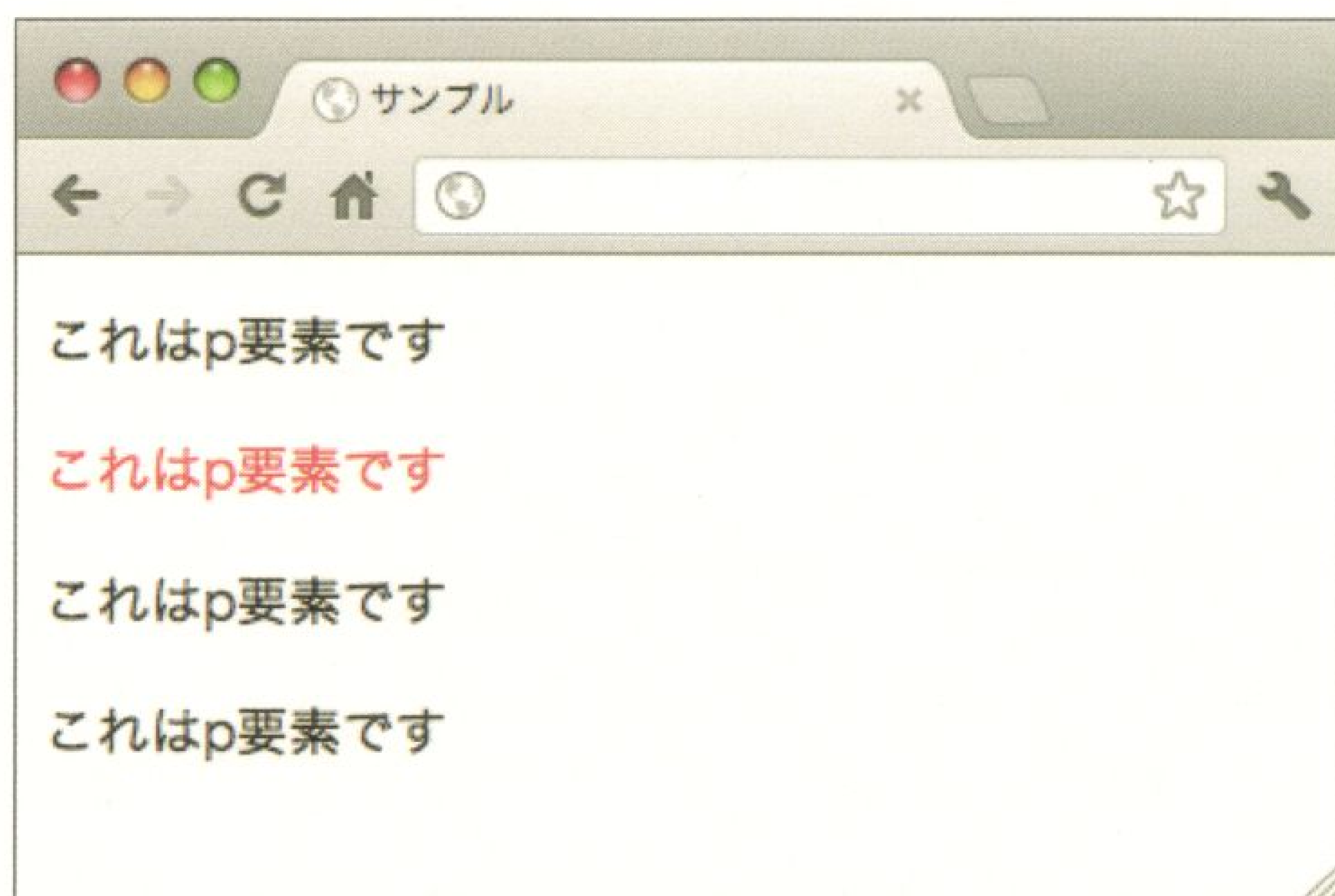
HTML sample/chapter-06/lecture-6-1/04.html

```
<p>これはp要素です</p>
<p id="abc">これはp要素です</p>
<p>これはp要素です</p>
<p>これはp要素です</p>
```

CSS sample/chapter-06/lecture-6-1/04-id.css

```
#abc { color: red; }
```

IDセレクタの使用例



上のソースコード例を表示させたところ

▶▶疑似クラス

疑似クラスとは、ある要素が特定の状態にあるときに限定して適用するセレクタです。たとえば、同じa要素でも「リンク先をまだ見ていない状態」と「リンク先をすでに見た状態」で異なる文字色を指定する場合などに使用します。「クラス」という名前がついてはいますが、class属性との関連はありません。

疑似クラスは全部で20種類以上ありますが、その多くはCSS3で新しく追加されたもので、ブラウザの対応状況などからその多くはまだ利用されていません。ここでは、一般的なWebページで利用されることの多い、以下の4つの疑似クラスを紹介しておきます。これらは古いブラウザでも基本的には問題なく利用できます。

疑似クラス	適用対象
要素:link	リンク先をまだ見ていない状態のa要素
要素:visited	リンク先をすでに見た状態のa要素
要素:hover	カーソルが上にある状態の要素
要素:active	マウスのボタンなどが押されている状態の要素

主要な4つの疑似クラス

「:link」と「:visited」はリンク部分、つまりa要素だけを対象としますが、「:hover」と「:active」はa要素以外にも使用できます。ただし、Internet Explorer 6以前のバージョンでは、「:hover」はa要素にしか適用されない点に注意してください。また、「:active」はブラウザの種類によっては反応しないものもあります。

上表の「要素」と書いてある部分には、クラスセレクタやIDセレクタのときと同じように「要素の名前」または「*」が指定でき、「*」は省略可能です。

疑似クラスの指定順序

一般に、リンク部分には表で示した4つの疑似クラスが同時に指定されます。そこで注意する必要があるのは、指定する順序です。
詳しくは「Lecture 6-4　指定が競合した場合の優先順位(p.122)」で解説しますが、CSSでは同じ部分に対して複数の異なる表示指定をすると、あとからの指定が有効となります(前の指定は後の指定で上書きされます)。

疑似クラスの中には、同時にその状態になり得るものとなり得ないものがあるのですが、同時にその状態になり得るもので指定が競合した場合、あとからの指定が有効になってしまいます。
たとえば、「:link」と「:visited」が同時にその状態になることはありませんが、「:hover」と「:active」はほかの状態と同時になることがあります。そのため、まず最初に「:link」と

「:visited」の指定を書き、そのあとにそれを上書きするように「:hover」と「:active」を順に指定してください[※9]。

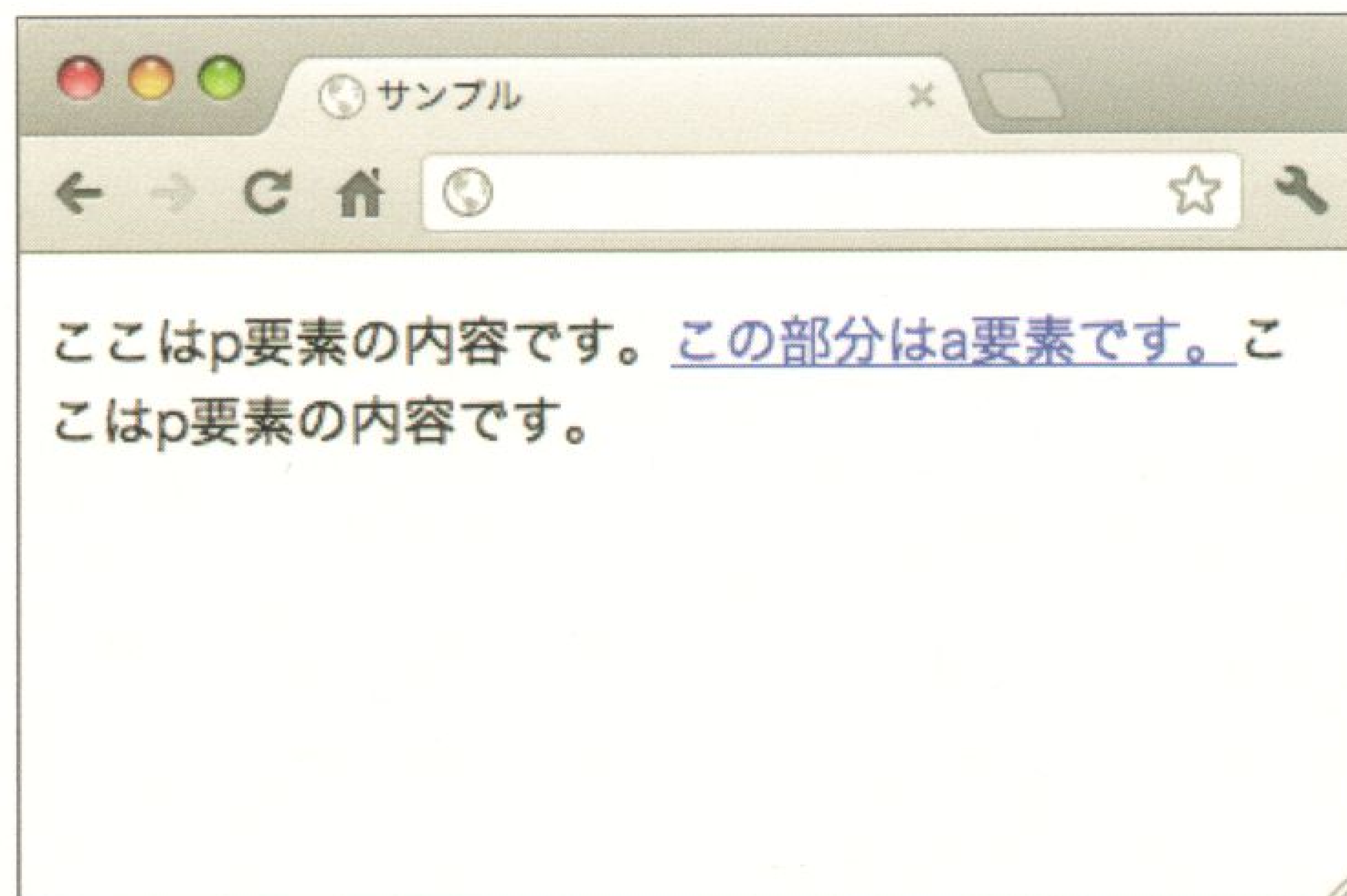
HTML　sample/chapter-06/lecture-6-1/05.html

```
<p>
ここはp要素の内容です。<a href="06.html">この部分はa要素です。</a>ここはp要素の
内容です。
</p>
```

CSS　sample/chapter-06/lecture-6-1/05-pseudo-class.css

```
a:link    { color: blue; }
a:visited { color: purple; }
a:hover   { color: red; }
a:active  { color: yellow; }
```

疑似クラスの使用例



上のソースコード例を表示させたところ

※9：たとえば、リンク部分の文字色の指定で、「:hover」のあとに「:link」と「:visited」の指定があると、「:hover」での指定は常に「:link」または「:visited」の指定に上書きされることになります。同様に、「:active」のあとに「:hover」があると、「:active」のときには常に「:hover」の指定で上書きされることになります。

▶▶ 結合子

セレクタをカンマ(,)で区切ることで、適用先を複数指定できることはすでに説明しました。それと同様に、セレクタを半角スペースで区切って複数並べると、「左側の適用対象」の中に含まれる「右側の適用対象」に表示指定が適用されます。半角スペースで区切るセレクタは2つに限らず、いくつでも区切って適用対象を絞り込むことができます。

たとえば「`h1 em`」と書くと、h1要素の中に含まれるem要素だけに適用されることになります(p要素内のem要素などには適用されません)。
同様に、「`body.top h1 em`」と書くと、「`class="top"`」が指定されているbody要素に含まれるh1要素にさらに含まれるem要素だけに適用されます。

HTML sample/chapter-06/lecture-6-1/06.html

```
<h1>これは<em>h1要素</em>です。</h1>
<p>これは<em>p要素</em>です。</p>
<p>これは<em>p要素</em>です。</p>
```

CSS sample/chapter-06/lecture-6-1/06-combinator.css

```
h1 em {
  color: red;
  font-style: normal;
}
```

結合子の使用例



上のソースコード例を表示させたところ

LECTURE 6-2

その他のセレクタ

では次に、まったく使われていないわけではないのですが、現状ではまだあまり使われていないセレクタを紹介します。これらのセレクタは、Internet Explorer以外のブラウザでは、極端に古いバージョンを除けばすべてに対応しています。しかし、Internet Explorerはバージョン9ですべてに対応したものの、それより前のバージョンでは未対応のものが多くあります。以降に示すセレクタの表では、Internet Explorer 6〜8の対応状況を○×で示してありますので参考にしてください。

▶▶ 属性セレクタ 【CSS3改】

属性セレクタは、特定の属性が指定されている要素、または特定の属性に特定の値が指定されている要素を適用対象とするためのセレクタで、次のものがあります。

属性セレクタ	適用対象	IE6	IE7	IE8
要素[属性名]	「属性名」の属性が指定されている要素	×	○	○
要素[属性名="属性値"]	「属性名」の属性に「属性値」の値が指定されている要素(値全体が一致)	×	○	○
要素[属性名~="属性値"]	「属性名」の属性に「属性値」の値が指定されている要素(値全体または半角スペース区切りの値のどれかと一致。クラスセレクタと同じ)	×	○	○
要素[属性名|="属性値"]	「属性名」の属性に「属性値」の値が指定されている要素(値全体またはハイフン区切りの値の前半が一致。言語コード「en-US」などに使用)	×	○	○
要素[属性名^="属性値の始め"]	「属性名」の属性の値が「属性値の始め」で始まる要素	×	○	○
要素[属性名$="属性値の終り"]	「属性名」の属性の値が「属性値の終り」で終わる要素	×	○	○
要素[属性名*="属性値の一部"]	「属性名」の属性の値が「属性値の一部」を含む要素	×	○	○

要素[属性名~="属性値"]という書式は機能的にはクラスセレクタとまったく同じですが、クラスセレクタはInternet Explorer 6でも対応しているのに対し、属性セレクタはInternet Explorer 6では未対応となっています。

▶▶ その他の疑似クラス 【CSS3改】

現時点ではまだそれほど多くは使われていない疑似クラスには、次のようなものがあります。

疑似クラス	適用対象	IE6	IE7	IE8
要素:nth-child(式)	先頭から○個目の要素から△個おきに適用	×	×	×
要素:nth-last-child(式)	最後から○個目の要素から△個おきに適用	×	×	×
要素:nth-of-type(式)	先頭から○個目の要素から△個おきに適用（同じ要素名の要素のみ対象）	×	×	×
要素:nth-last-of-type(式)	最後から○個目の要素から△個おきに適用（同じ要素名の要素のみ対象）	×	×	×
要素:first-child	子要素の中で最初の要素	×	○	○
要素:last-child	子要素の中で最後の要素	×	×	×
要素:first-of-type	子要素の中で最初の要素（同じ要素名の要素のみ対象）	×	×	×
要素:last-of-type	子要素の中で最後の要素（同じ要素名の要素のみ対象）	×	×	×
要素:only-child	唯一の子要素である場合に適用	×	×	×
要素:only-of-type	唯一の子要素である場合に適用（同じ要素名の要素のみ対象）	×	×	×
要素:focus	フォーカス（選択）された状態の要素	×	×	○
要素:checked	ラジオボタンやチェックボックスがチェックされた状態の要素	×	×	×
要素:disabled	「disabled」の状態の要素	×	×	×
要素:enabled	「disabled」の状態ではない要素	×	×	×
要素:root	ルート要素（html要素）	×	×	×
要素:empty	要素内容が空の要素	×	×	×
要素:target	URLの最後が「#○○○」となっているリンクをクリックした時の対象要素	×	×	×
要素:lang(言語コード)	「言語コード」の言語に設定されている要素	×	×	○
要素:not(セレクタ)	「セレクタ」の対象外のすべての要素	×	×	×

ここで、表の上から4つの疑似クラスにある「式」について説明しておきましょう。
この式は「○個目から△個おき」という情報を示すためのもので、基本的には「an+b」という形式の書式であらわします。この書式では、「n」は0から1ずつ増える整数としてそのまま使い、「a」と「b」の部分に具体的な整数を当てはめて使用します。「a」と「b」には負の整数や0も指定でき、値が0となる場合には表記を省略することも可能です（ただし省略できるのは「a」と「b」のいずれか一方のみ）。

たとえば、「:nth-child(2n+1)」のように指定すると、最初は「n」は0ですので「2×0+1＝1」、次は「n」は1となって「2×1+1＝3」、その次は「n」は2となって「2×2+1＝5」、というように奇数個目（1個目から2個おき）の要素に適用されるようになります。
同様に、偶数個目に適用したい場合は「:nth-child(2n+0)」または単に「:nth-child(2n)」のように書くことができます。
また、「an」部分を省略して「b」だけを指定し、「:nth-child(3)」のように指定すると3個目、「:nth-child(5)」のように指定すると5個目の要素だけに適用することができます。

この書式の数値を変えることで「○個目から△個おき」の要素に自由に適用が可能となりますが、単純に「奇数個目」「偶数個目」をあらわす場合には、式を使わずに「odd」「even」というキーワードも指定できるようになっています。これらのセレクタは、表の色を1行置きに変えたい場合などに使用すると便利です。

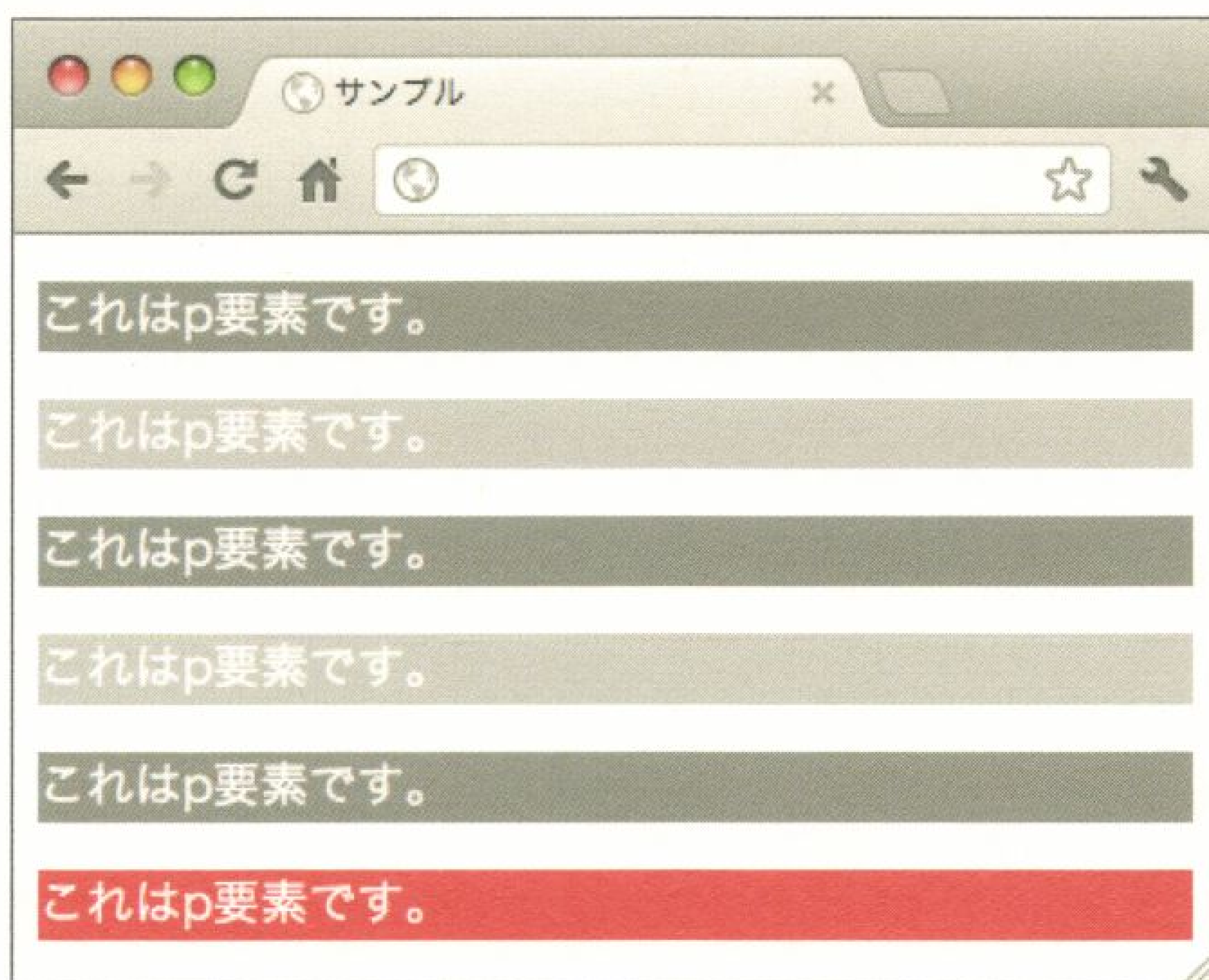
HTML　sample/chapter-06/lecture-6-2/01.html

```html
<p>これはp要素です。</p>
<p>これはp要素です。</p>
<p>これはp要素です。</p>
<p>これはp要素です。</p>
<p>これはp要素です。</p>
<p>これはp要素です。</p>
```

CSS　sample/chapter-06/lecture-6-2/01-nth-child.css

```css
p:nth-child(odd) {                /* 奇数個目 */
  color: white;
  background-color: gray;
}
p:nth-child(even) {               /* 偶数個目 */
  color: white;
  background-color: silver;
}
p:nth-child(6) {          /* 6個目 */
  color: white;
  background-color: red;
}
```

:nth-child疑似クラスの使用例



前ページのソースコード例を表示させたところ

▶▶ 疑似要素 【CSS3改】

疑似要素とは、簡単に言えばタグのつけられていない範囲（つまりHTMLの要素にはなっていない部分）に対してCSSを適用するためのセレクタです。たとえば、ある要素の1文字目だけにCSSを適用したり、HTML上では存在していないコンテンツを追加してその部分の表示指定をすることもできます。CSSでコンテンツを追加する方法については、Chapter 10で解説しますが、疑似要素に分類されるセレクタには次のものがあります。

疑似要素	適用対象	IE6	IE7	IE8
要素::first-line	ブロックレベル要素の1行目	○	○	○
要素::first-letter	ブロックレベル要素の1文字目	○	○	○
要素::before	要素の直前にコンテンツを追加	×	×	○
要素::after	要素の直後にコンテンツを追加	×	×	○

これらの疑似要素のうち、「::first-line」と「::first-letter」に関しては、適用できる要素がブロックレベル要素に限定されている点に注意してください。

実はCSS2.1までは、疑似要素も疑似クラスと同じく「:first-line」「:first-letter」と書くことになっていました。CSS3になってから、疑似クラスと明確に区別できるようにする目的で、疑似要素にはコロンを2つつけることになったのです。そのため、CSS2.1との互換性を維持する目的で、ブラウザはコロンを1つしかつけていない疑似要素にも対応することになっています。

▶▶ その他の結合子 【CSS3改】

Internet Explorer 6が未対応だったため、これまではそれほど使われてこなかった結合子には、次のものがあります。

結合子	適用対象	IE6	IE7	IE8
セレクタ1 > セレクタ2	「セレクタ1」の直接の子要素である「セレクタ2」の要素	×	○	○
セレクタ1 + セレクタ2	共通の親要素を持つ要素の中で「セレクタ1」の直後にあらわれる「セレクタ2」の要素	×	○	○
セレクタ1 ~ セレクタ2	共通の親要素を持つ要素の中で「セレクタ1」よりも後にあらわれる「セレクタ2」の要素	×	○	○

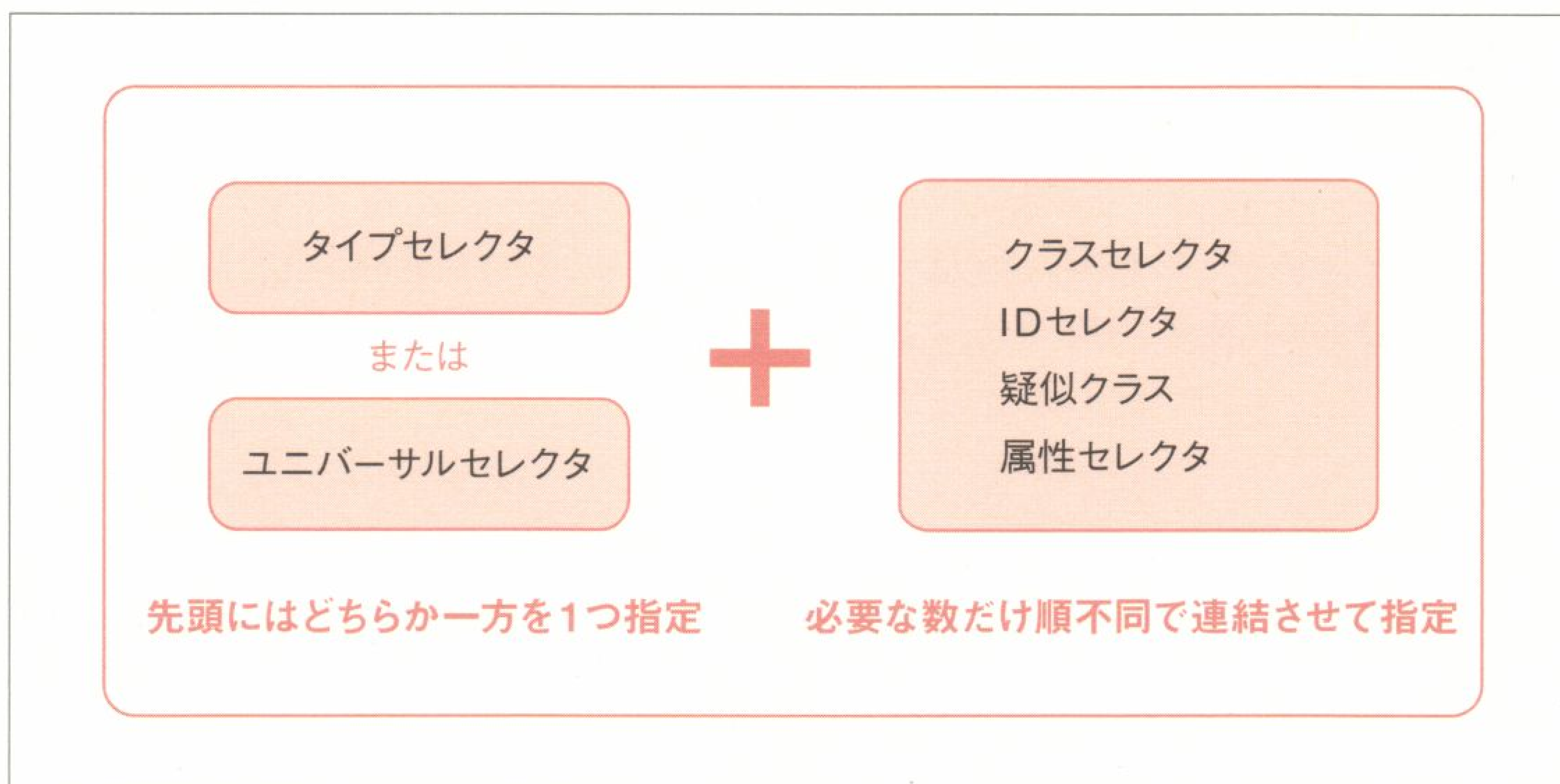
LECTURE 6-3

セレクタの組み合せ方

セレクタは単独で使うだけではなく、組み合わせて使うこともできます。組み合わせ方によってさまざまな指定が可能です。

▶▶ セレクタの基本単位

セレクタの種類が一通り分かったところで、次にそれらの組み合せ方のルールについて説明します。はじめはセレクタの基本単位（最小構成）となる部分の組み合わせのルールからです。セレクタの基本単位は、疑似要素と結合子以外のセレクタで構成されます。
基本単位の先頭には、かならずタイプセレクタ（要素名）またはユニバーサルセレクタ（*）のいずれかを1つ配置する必要があります。もちろん、そのあとに何かが続く場合はユニバーサルセレクタを省略できます。あとはそれに続けてクラスセレクタ・IDセレクタ・疑似クラス・属性セレクタのうちの必要なものを必要な数だけ順不同で連結させたものがセレクタの基本単位となります。



基本単位の組み合せ方

セレクタの基本単位は、結合子で区切っていくつでも指定できます。結合子の前後には、半角スペース・改行・タブを自由に入れることができます。疑似要素は、セレクタ全体の最後尾に1つだけ指定することができます。

LECTURE 6-4

指定が競合した場合の優先順位

さて、実際の制作においてCSSによる長い表示指定をしていると、どこかですでに表示指定をしているにもかかわらず重複する指定をしてしまうことがあります。これは複数人によるチームでCSSを指定している場合や途中で担当者が変わった場合などによく起こります。そのようなとき、同じ適用先に対して異なる表示指定があると、どうなってしまうのでしょうか？実際には、CSSには競合した指定の優先順位を決めるためのルールがあり、それにはセレクタも大きく関係しています。ここでは、そのルールについて詳しく説明しておきます。

▶▶ 優先順位の決定方法

同じ適用対象に対して、複数個所で異なる表示指定がされた場合、CSSでは基本的に次のルールで優先される指定を決定し適用します。

1. !important のついている指定は最優先
2. 使用しているセレクタの種類から優先度を計算
3. 計算結果の優先度が同じなら後の指定を優先

まず、CSSの宣言（「プロパティ名: 値」の部分）のうしろに「!important」と書いておくと、その指定が最優先されます。たとえば、次の例の3つの中では、真ん中の指定が優先されることになります。

```
p { color: red; }
p { color: green !important; } ←―― この指定が優先される
p { color: blue; }
```

「!important」がつけられている指定が優先して適用される

「!important」がつけられていなかったり、「!important」が複数個所につけられている場合は、その優先順位をセレクタの種類から導き出すことになります。これについては、次の「セレクタからの優先度の計算方法」で、その計算方法を詳しく説明します。

セレクタで計算しても結果的に優先順位が同じになってしまった場合は、より後からの指定の方が

優先されることになります(つまり、優先順位が同じであれば、後からの指定が前からの指定を上書きするということです)。

COLUMN

「!important」はユーザースタイルシートでも使用できる

Webページを制作する際にはそれほど気にする必要もないのですが、CSSはWebページの制作者だけでなく、ユーザーからも指定できるようになっています(多くのブラウザはユーザーが指定したCSSファイルを読み込めるようになっており、そのようなCSSファイルはユーザースタイルシートと呼ばれています)。さらに、制作者が特にCSSを指定しなくても、ブラウザはHTMLをそれなりに内容が分かるように表示しますが、それはブラウザがデフォルトのCSS(または結果的にそのようになる仕組み)を持っているからです。つまり、CSSは「制作者」「ユーザー」「ブラウザ」という三者のあいだでも競合する可能性があるということです。その三者における優先順位は次のようになっています。

1. ユーザーのCSS(!importantつき)
2. 制作者のCSS(!importantつき)
3. 制作者のCSS
4. ユーザーのCSS
5. ブラウザのデフォルトCSS

実は「!important」はユーザーのCSSでも使用でき、ユーザースタイルシート内でそれを使った場合がもっとも優先度が高くなります。これはたとえば視覚障害のある人の「背景を暗く、文字色を明るくしないと見えにくい」といったニーズに応えることができるようにするためです。
「!important」を使わない場合は、制作者のCSSが最優先で適用されます。

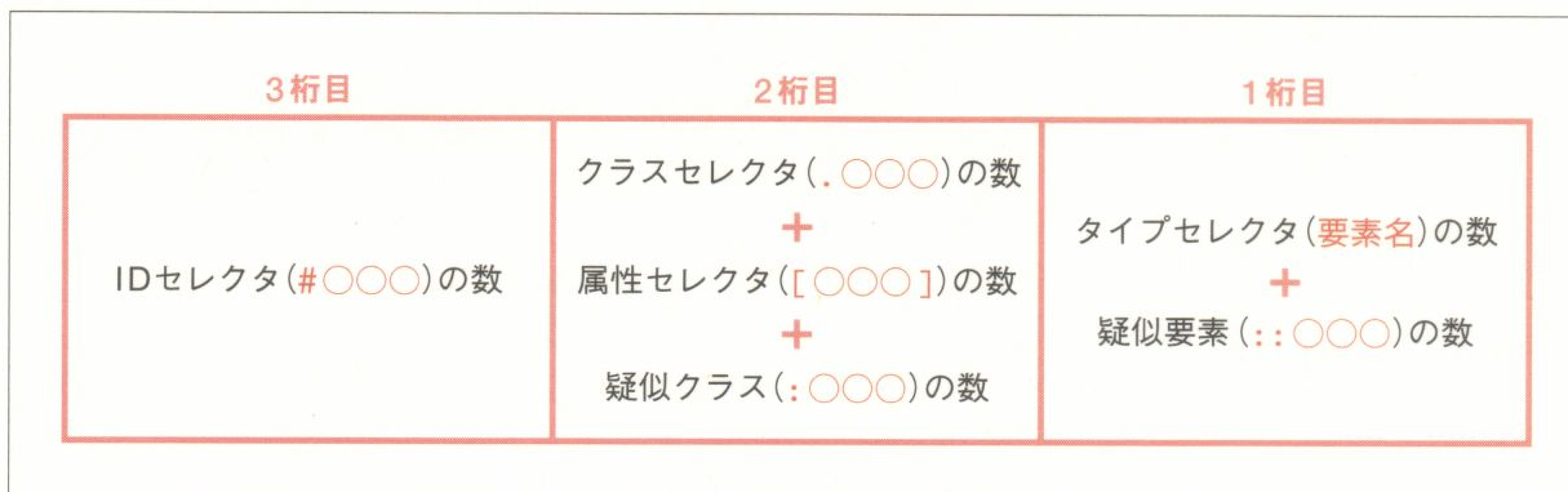
CHAPTER 6

▶▶ セレクタからの優先度の計算方法

続けて、セレクタから優先順位を計算する方法を説明しましょう。しかしその前に、CSSにはセレクタのない指定があったことも思い出してください。それはstyle属性の値としてCSSの宣言部分だけを指定する場合です。実はこのセレクタのない指定が、セレクタによる優先順位としてはもっとも高くなります。

style属性による指定ではなく、セレクタが指定されている場合の優先順位の計算方法は次ページの図の通りです。簡単に言えば、セレクタを構成する各部分を種類別に分類して数を数え、それ

によって作られた3桁[※10]の数字が大きい方が優先順位が高くなる、ということです。見ると分かるように、3桁目はIDセレクタ、2桁目はid属性以外の属性関連セレクタ、1桁目は要素関連セレクタとなっていて、IDセレクタを使うと優先度が高くなることが分かります。



セレクタから優先順位を計算する方法。この3桁の数字が大きいほど優先順位は高くなる

なお、この計算においては、ユニバーサルセレクタ(*)は無視されます。また、疑似クラスのうち「:not()」は、()内のセレクタはほかの部分と同じようにカウントされますが、「:not()」自身は疑似クラスとしてはカウントされません。

※10：この3桁の数字の各桁は、たとえ該当するセレクタが10個以上あった場合でも、決して繰り上がらないものとして考えることになっています(つまり、普通の10進数ではないということです)。したがって、2桁目に該当するクラスセレクタがたとえ100個あったとしても、IDセレクタが1つ入っているセレクタの方が優先度は高くなります。

CHAPTER 7

ページ内の構造

「ページ全体の枠組み」とその内容となる「テキスト」についてはすでに学習しましたので、ここではページを構成する各部分の枠組み・領域を作るための要素とプロパティを紹介していきます。また、テキスト以外のコンテンツ(画像・動画・音声など)を埋め込むための要素についてもここで学習します。

LECTURE 7-1

基本構造を示す要素

まず最初に、ページ内の基本的な構造をつくる要素として、章、節、項のような大きな枠組みの範囲を表す要素や、ヘッダー、フッターのような領域を示すための要素などを紹介します。

セクションについて

HTML5よりも前のHTML・XHTMLにおいては、見出しをあらわす要素（h1～h6）は用意されていましたが、章・節・項のような範囲を明確に示すための要素やルールは用意されていませんでした。そのため、章や節などの範囲は、見出しの階層などから推測することしかできませんでした。

HTML5からは、章・節・項などの範囲（セクション）を示すための新しい要素とルールが導入されています。セクションの範囲を示す要素を使用すると、セクションの範囲が明確に分かるだけでなく、その親子関係（「章」の中に別セクションがあれば「節」と判断できるなど）も明確になります。また、必ずしもセクションのタグがつけられていなくてもセクションの範囲が分かるように、次のようなルールも導入されました。

▶▶ セクション見出しのルール

まず、セクションの先頭にある見出しは、そのセクションの見出しとなります。そして、次の見出しがあらわれたときに、**見出しの階層が前の見出しと同じかそれより上であればその直前で前のセクションは終了し、新しい見出しとともに新しいセクションがはじまっているものと判断します。見出しの階層が前の見出しよりも下であった場合は、そこから（前のセクションに含まれる）サブセクションがはじまるものと判断**します。

このようなルールができたため、たとえば見出しを主題と副題のようにわけてh1要素とh2要素を連続して使ったような場合でも、2つのセクションが開始されることになってしまいます。Chapter 5で登場したhgroup要素は、そのような場合でもセクションが1つしか開始されないようにするために、見出しをグループ化できるようにする目的で用意されたものなのです。

▶▶ セクションと見出しの使い分け

このような新しい要素とルールの導入により、セクション関連要素を適切に使用していれば見出しのレベルに関係なく章・節・項などの親子関係が分かり、逆にセクション関連要素を使っていなくても適切な階層の見出しを使用していればセクションの範囲と親子関係を明確に示すことができるようになりました。しかし、セクションの範囲にはセクションを示すタグを明示的につけ、見出しについてはそれぞれの階層に合わせた適切なレベルの見出しを使うか、またはh1要素だけを使用することが推奨されています。

セクションをあらわす要素 【HTML5新】

HTML5では、セクションの範囲を示す要素は全部で4種類ありますが、ここではそのうちの3つを紹介します(残りの1つはナビゲーションのセクションを示すための専用要素で詳しくは「Chapter 8　ナビゲーション」で解説します)。

要素名	意味
section	セクション
article	内容がそれだけで完結しているセクション
aside	主コンテンツには含まれないセクション

HTML5のセクションをあらわす要素

section要素は、その範囲が一般的な章・節・項など(つまり普通のセクション)であることを示す要素です。**article要素**は、セクションの中でもそれだけで完結している(独立した)ものに対して使用されます。たとえば、新聞や雑誌などの記事、ブログの記事、ユーザーによるブログ記事へのコメントなどがこれに該当します。**aside要素**は、そのページのメインコンテンツには含まれない補足記事、広告、一部抜粋を使った記事の紹介文などの範囲に対して使用されます。

基本構造を示すその他の要素

セクションの範囲をあらわすわけではありませんが、ページ内(セクション内)で基本構造を示す要素には、次のものがあります。

要素名	意味
header	ヘッダー
footer	フッター
address	問い合わせ先

これらもHTML5の基本構造を示す要素ではあるが、その範囲はセクションにはならない

▶▶ header要素 【HTML5新】

header要素は、いわゆるヘッダー的な部分の範囲を示すために使用される要素です。一般的には、セクションの見出しのほか、導入文的なテキストやロゴ画像、検索フォーム、セクション内の目次などを含む範囲に対して使用されます。この要素内には、ほかのheader要素およびfooter要素を入れることはできませんので注意してください。

▶▶ footer要素 【HTML5新】

footer要素は、いわゆるフッター的な部分の範囲を示すために使用される要素です。この要素を囲うもっとも近いセクションのフッターとなります。一般的には、記事を書いた人の名前や関連する内容へのリンク、Copyrightのテキストなどを含む範囲に対して使用されます。索引や奥付、使用許諾、Appendixなどの場合は、セクションの内容全体がfooter要素となる場合があります。多くの場合、この要素はセクション内の最後に配置されますが、必ずしもその必要はなく、セクション内の最初に配置してもかまいません。この要素内にも、ほかのheader要素およびfooter要素を入れることはできない点に注意してください。

▶▶ address要素

address要素は、多くの場合footer要素内で使用され、その部分が問い合わせ先（連絡先）であることを示します。article要素内に入れられている場合はそのセクションに関する問い合わせ先となり、そうでなければページ全体の問い合わせ先となります。要素名はaddressですが、**住所を示すための要素ではない**点に注意してください（もちろん問い合わせ先の1つとして住所を入れてもかまいませんが、単純に住所部分をマークアップするだけならp要素を使用してください）。また、Copyrightのテキストや更新日などの情報は問い合わせ先ではありませんので、address要素内には入れないでください。この要素内に入れられない種類の要素としては、header要素・footer要素・address要素のほか、見出しとセクションがあります。

```
<footer>
  <address>
    お問い合わせ先：
    <a href="mailto:info@example.jp">山田</a>
  </address>
  <p>
    <small>&copy; copyright 2012 Example.</small>
  </p>
</footer>
```

footer要素とaddress要素の使用例

COLUMN

HTML5の新要素を古いIEに認識させる方法

実は、HTMLで新しく追加されたsection要素やarticle要素、aside要素、header要素、footer要素といった要素は、Internet Explorerのバージョン8以前では要素として認識されません。そのため、それらの要素にCSSを指定しても一切適用されないのです。

しかし、JavaScriptを使用して新しい要素ごとに「document.createElement('要素名');」のように指定すると、古いInternet Explorerでも新要素が認識され、CSSも適用されるようになります。このようなJavaScriptのソースコードはもちろん自分で書いてもOKなのですが、多くの人が何度も検証して修正を重ねてきた問題のないオープンソースのソースコードが「http://code.google.com/p/html5shiv/」で公開されており、自由にダウンロードして使えるようになっています。使い方は、ダウンロードしたファイルの中に含まれている「html5shiv.js」をサーバーにアップロードし、次のソースコードをHTMLファイルのhead要素内に書いておくだけでけです(青で示した部分は「html5shiv.js」を入れた階層までのパスに変更してください)。なお、このファイルの使用方法は変更されることがありますので、実際に使用する場合はサイトに掲載されている使用方法を確認してください。

```
<!--[if lt IE 9]>
<script src="～/html5shiv.js"></script>
<![endif]-->
```

「html5shiv.js」をサーバーに入れ、このソースコードをhead要素内に書いておくだけで、古いIEでもHTML5の新要素を認識するようになる

CHAPTER 7

LECTURE 7-2

画像・動画・音声関連要素

構造や枠組みをあらわす要素ではありませんが、ここで画像や動画、音声などのファイルをWebページ内に組み込むための要素も紹介しておきます。

画像

▶▶ img要素

はじめは、Webページに**画像を組み込むためのimg要素**からです。この要素は、要素内容のない空要素で、表示させたい画像のURLをsrc属性で指定すると画像が表示されるようになります。さまざまな形式の画像を表示させられますが、一般的にはPNG形式、JPEG形式、GIF形式が使用されています。

▶▶ alt属性について

alt属性には、画像が正しくロードできなかった場合や、そもそも画像を表示することができない音声や点字などの環境で利用する、**画像の代わりのテキスト**を指定します。ここで指定するのは、あくまで"画像の代わり"として使用するテキストであって、画像の説明をするためのテキストではない点に注意してください(つまり、画像が表示されない状態でそれがどんな画像なのかを説明するテキストではなく、画像の代わりとして機能するテキストを入れるということです)。画像に関する説明や補足情報などを含めたい場合には、グローバル属性のtitle属性を使用します。

HTML5よりも前のHTML/XHTMLでは、alt属性を指定することは必須とされていました。しかし、HTML5では不要と考えられる場面においては、img要素にalt属性をつける必要はありません。

img要素に指定できる属性

・**src="画像のURL"**　※必須
表示させる画像のURLを指定します。

・**alt="代替テキスト"**
画像が表示できない場合に、その代わりとして使用するテキストを指定します。

・**width="幅"**
画像の幅（実際の幅ではなく表示させる幅）をピクセル数（単位をつけない整数）で指定します。

・**height="高さ"**
画像の高さ（実際の高さではなく表示させる高さ）をピクセル数（単位をつけない整数）で指定します。

```
<h1>
<img src="logo.gif" alt="株式会社マイナビ" width="250" height="200">
</h1>
```

img要素の使用例



img要素の表示例

動画と音声 HTML5新

▶▶ video要素とaudio要素

次に、動画を組み込む**video要素**と音声を組み込む**audio要素**について説明します（source要素についてはこのあとに解説します）。両者には共通する属性も多く、動画や音声のデータは**src属性**で指定します。要素内容には、これらの要素に未対応のブラウザで表示させるメッセージを入れま

す(したがって、video要素・audio要素に対応したブラウザでは要素内容は表示されないようになっています)。

要素名	意味
video	動画を組み込む
audio	音声を組み込む
source	動画や音声の代替データを指定する

動画と音声を組み込むための要素

video要素とaudio要素に指定できる属性の中には、「属性名="属性値"」の形式をとらず、「属性名」だけで指定できるものがあります。そのような属性のことを**論理属性**と言います。論理属性の値は論理値「true」または「false」のいずれかで、属性を指定すると「true」になり、指定しなければ「false」となります。ちょっと難しい話になってきましたが、分かりやすく言えば「true」と「false」は「Yes」と「No」のようなもので、**controls属性**を指定すればコントローラーが表示され、**loop属性**を指定すればループ再生されるようになります。それらを指定しなければ、コントローラーは表示されず、ループ再生もされない状態となります。

論理属性は「controls」のように属性名だけで指定する形式のほかに、「controls="controls"」のように属性値に属性名をそのまま入れても有効ですし、「controls=""」の要に値を空にして指定することもできます。

video要素に指定できる属性

- **src="動画のURL"**
  表示させる動画のURLを指定します。

- **width="幅"**
  動画の幅(実際の幅ではなく表示させる幅)をピクセル数(単位をつけない整数)で指定します。

- **height="高さ"**
  動画の高さ(実際の高さではなく表示させる高さ)をピクセル数(単位をつけない整数)で指定します。

- **poster="画像のURL"**
  動画が再生できるようになるまでの間に表示させておく画像のURLを指定します。

- **controls**
  動画の再生・ポーズ・ボリューム調整などをおこなうコントローラー部分を表示します。

- **autoplay**
  途中で止まらずに再生を続けられる程度のデータを読み込んだ段階で、自動的に再生を開始します。

video要素に指定できる属性（続き）

・loop
　動画をループ再生します。

・muted
　音を消した状態で再生します。

audio要素に指定できる属性

・src="音声データのURL"
　音声データのURLを指定します。

・controls
　音声データの再生・ポーズ・ボリューム調整などをおこなうコントローラー部分を表示します。

・autoplay
　途中で止まらずに再生を続けられる程度のデータを読み込んだ段階で、自動的に再生を開始します。

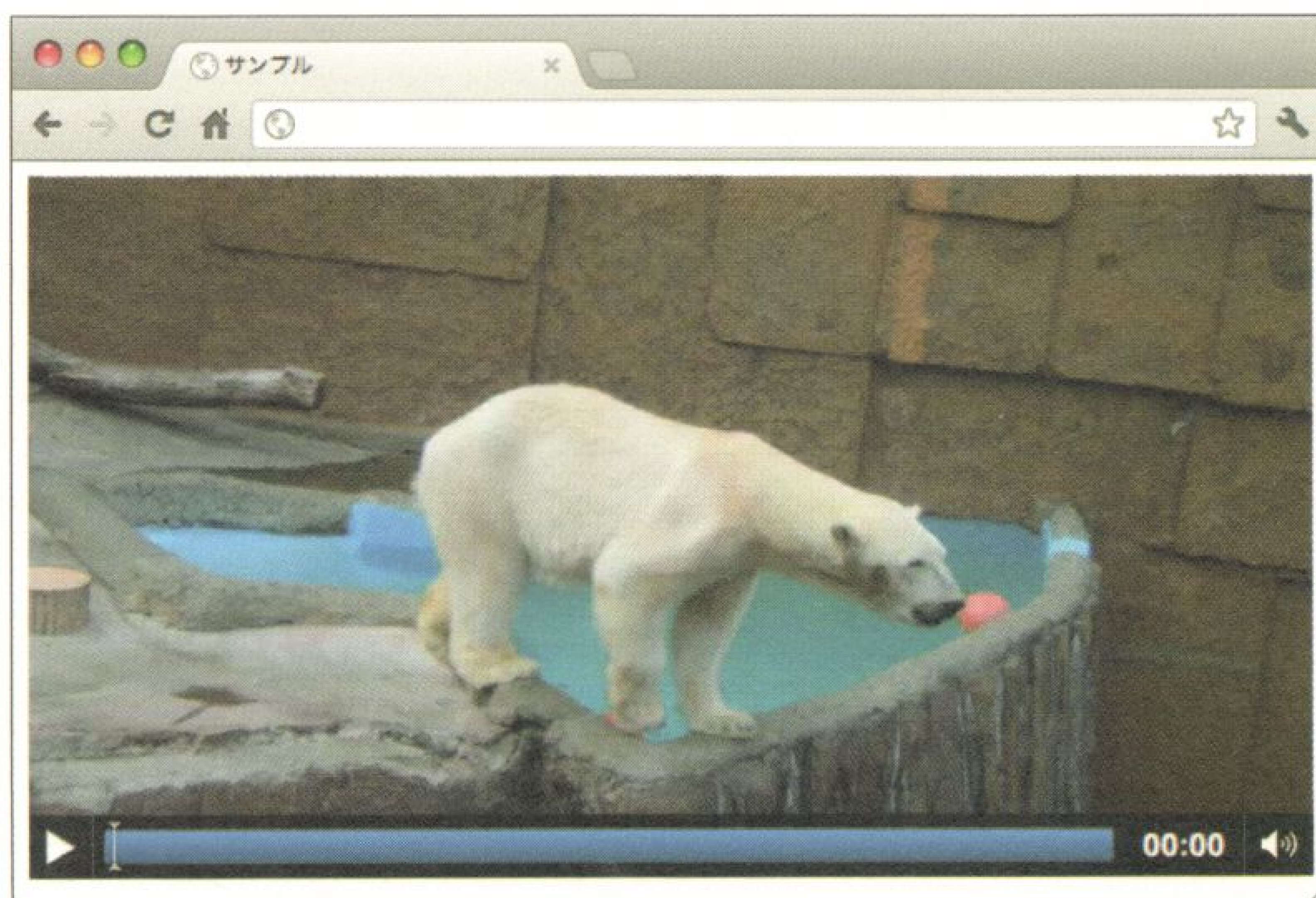
・loop
　音声データをループ再生します。

・muted
　音を消した状態で再生します。

sample/chapter-07/lecture-7-2/01.html

```
<video src="bear.mp4" controls>
  <p>HTML5に対応したブラウザでは動画が再生できます。</p>
</video>
```

video要素の使用例



video要素の表示例

▶▶ source要素

video要素またはaudio要素のsrc属性でデータをする場合は、1つのデータしか指定できません。予備としてほかの形式のデータも指定しておきたい場合には、video要素またはaudio要素のsrc属性を使用せずに、要素内容として**source要素**を組み込んでデータを指定します。source要素はいくつでも指定でき、より先に指定されているデータで再生可能なものが再生されます。source要素は要素内容のない空要素で、video要素またはaudio要素内のその他のコンテンツよりも前に配置する必要があります。

source要素に指定できる属性

・**src="データのURL"**　※必須
　再生するデータのURLを指定します。

・**type="MIMEタイプ"**
　再生するデータのMIMEタイプを指定します。

sample/chapter-07/lecture-7-2/02.html

```
<video controls>
  <source src="bear.mp4" type="video/mp4">
  <source src="bear.ogv" type="video/ogg">
  <p>HTML5に対応したブラウザでは動画が再生できます。</p>
</video>
```

source要素の使用例

LECTURE 7-3

ボックス関連プロパティ

さて、ここから話はCSSへと変わりますが、ここから説明する内容はCSSの中でも主要な部分を構成するものです。登場するプロパティの数が少し多めですが、整理して考えると基本的な部分では意外とシンプルなパターンに沿って作られていますので、1つずつしっかりと覚えていきましょう。

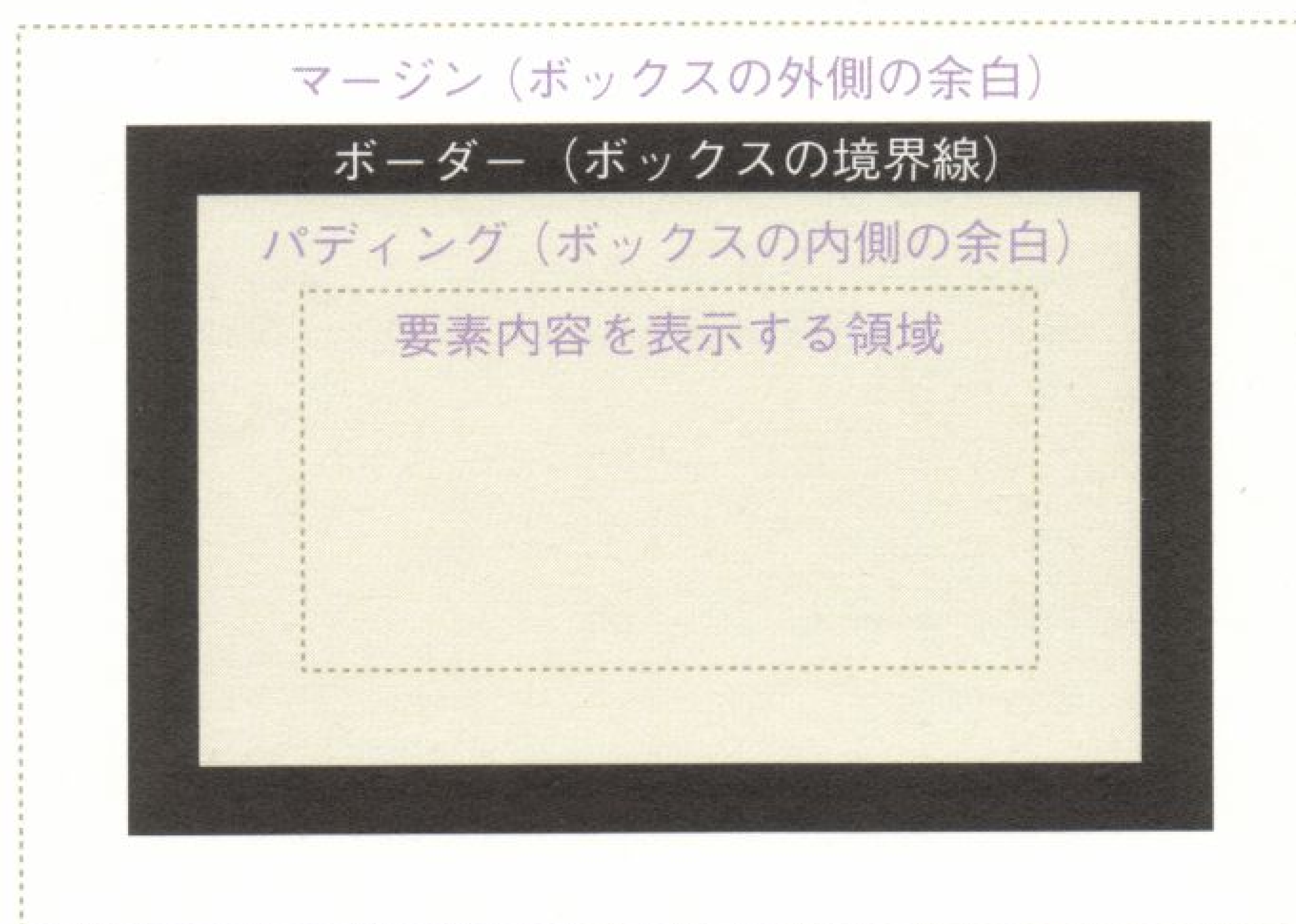
ボックスとは？

ここまでの解説やサンプルでは、たとえば背景の指定をする場合でも、それが表示される領域の範囲や境界については具体的に触れませんでした。ここではまず、各要素に確保される表示領域とその境界線、余白について説明し、そのあとでそれらを調整するために大量に用意されたプロパティ[※11]を紹介していきます。

CHAPTER 7

▶▶ ボックスの構造

HTMLの各種要素には、改行をおこなうためのbr要素のような一部の例外を除き、ボックスと呼ばれる四角い領域が用意されます。すべてのボックスは共通して、右図のような構造になっています。



ボックスの構造

※11：ここで紹介するボックス関連のプロパティは、全部で50近くあります。とはいっても、たとえば余白でも上用・下用・左用・右用・上下左右の一括指定用というように同じ機能で細かく別れているために数が多くなっているだけですので安心してください。

まず、ボックスには要素内容（テキストなど）を表示するための領域が用意されます。図ではもっとも内側の点線で示された部分です。そして、そのまわりには**ボーダー**と呼ばれる境界線を表示させることができるのですが、その内側と外側の両方に余白をとることができます。ボーダーの内側の余白は**パディング**と言い、外側の余白は**マージン**と言います。これらの境界線と余白は、上下左右別々にその幅を指定することができます。実は、CSSで指定した背景が表示されるのは、この**ボーダーの領域まで**で、マージン部分は常に透明となります。ボーダーは常に背景の上に表示されるため、ボーダーの領域に背景が表示されるのはボーダーを透明や半透明にしたときや、点線のように線の間に隙間がある線種に設定した場合などに限られます。

ボックスの初期状態

なお、各種要素のボックスの初期状態は、要素の種類やブラウザの種類によって異なります。たとえば、一般的なブラウザでは、p要素には最初から一行ぶん程度の上下のマージンが設定されていますが、div要素のマージンはまったく無い状態となっています。このように要素やブラウザの種類によってボックスの初期状態が違っているのは、Chapter 6で説明した「ブラウザのデフォルトCSS」(p.123参照）というものがあるからです※12。

マージン

マージンを設定するプロパティには、上下左右をそれぞれ個別に設定するものと、上下左右を一括して設定するものがあります。
このあとに紹介するパディング用のプロパティなども、同様のパターンのものが用意されています。

プロパティ名	定対象	指定できる値の数
margin-top	上のマージン	1
margin-bottom	下のマージン	1
margin-left	左のマージン	1
margin-right	右のマージン	1
margin	上下左右のマージン	1～4

マージンを設定するプロパティ

※12：このようなブラウザの種類による初期状態の違いをなくす目的で用意されたCSSのことを「リセットCSS」と呼び、多くのサイトのCSSのソースコードの先頭にはそのような指定が書かれています。ただし、「リセットCSS」は必須ではなく、その内容もサイトによって異なります。

各プロパティに指定できる値は次の通りです。

たとえば、「margin-top: 50px;」と指定すると上のマージンが50ピクセルになります。マージンを「auto」にすると、ボックスの他の領域のサイズなどからマージンが自動的に割り出されます。ボックスの幅を固定して左右のマージンを「auto」にすると、左右のマージンは同じ距離になり、結果としてボックスはセンタリングされます。

マージン関連のプロパティに指定できる値

- **単位つきの実数**
  マージンを単位つきの実数(30pxなど)で指定します。

- **パーセンテージ**
  この要素を含んでいるブロックレベル要素の幅(要素内容を表示する領域の幅)に対するパーセンテージで指定します(25%のように、実数の直後に半角の「%」をつけて指定します)。上下のマージンについても、幅に対するパーセンテージとなります。

- **auto**
  マージンをボックスの状況から自動的に設定します。

▶▶ marginプロパティの指定方法

上下左右のマージンを一括して指定するmarginプロパティだけは、値を半角スペースで区切ることで最大4つまで指定できます。指定した値の数とそれらの値がどのマージンに適用されるのかは、以下の表のようになっています。値を1つだけ指定した場合はそれが上下左右に適用され、値を4つ指定するとそれが上から時計回りに右・下・左と適用されます。このあとに登場するパディングなどの上下左右を一括で指定するプロパティも同じパターンになっていますので、値の数と適用される上下左右の関係はしっかりと覚えてください。

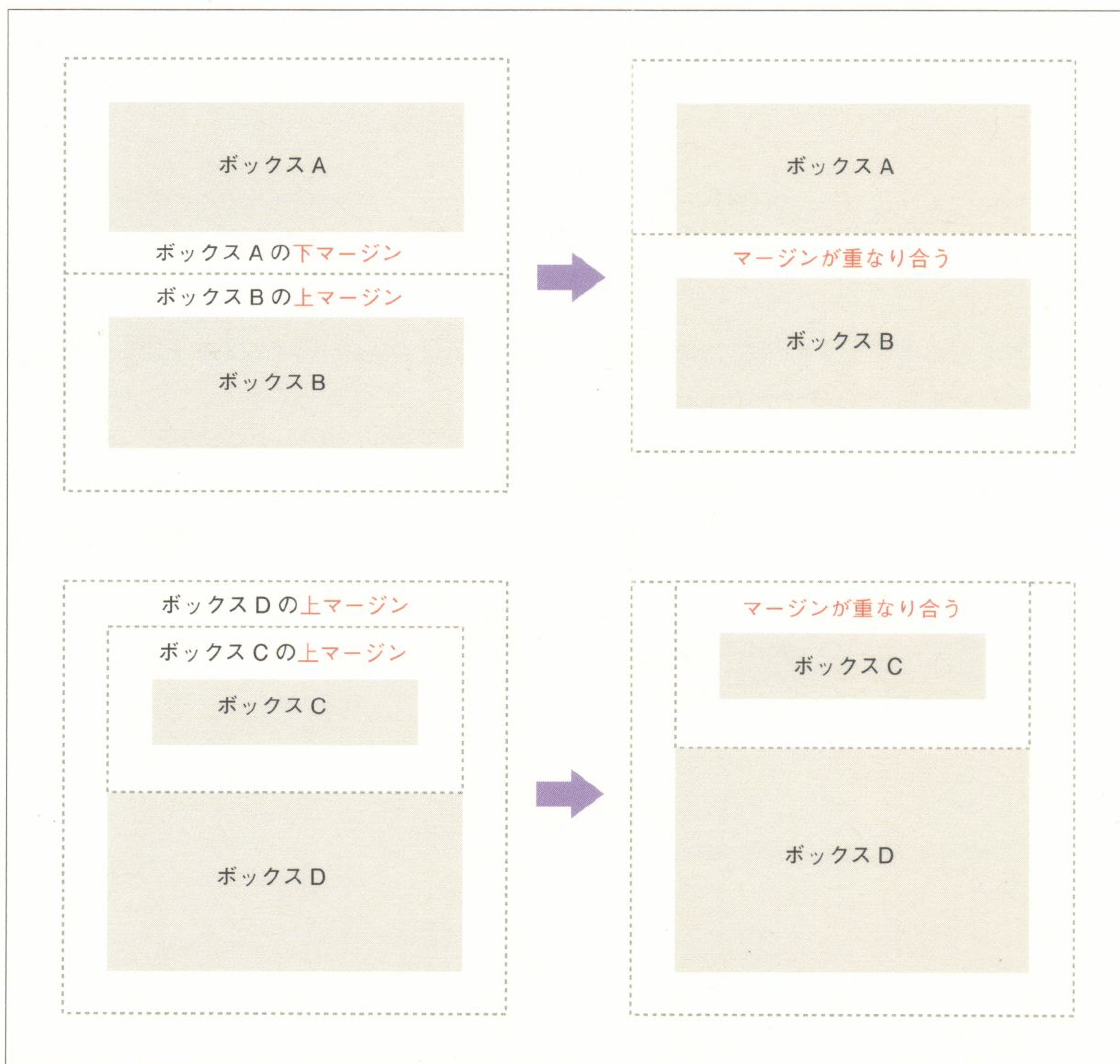
値の数	各値の適用場所	指定例
1	上下左右	margin: 10px;
2	上下　左右	margin: 10px 20px;
3	上　左右　下	margin: 10px 20px 30px;
4	上　右　下　左	margin: 10px 20px 30px 40px;

上下左右を一度に設定できるプロパティの値の指定パターン

CHAPTER 7

▶▶ マージンが隣接している場合の注意

また、マージンに関しては、特に覚えておいて欲しいルールがあります。それは、ボックスの上下のマージンは、ほかのボックスの上下のマージンと隣接している場合は、それらは重なり合うということです。たとえば、上下のマージンが10ピクセルずつに設定されているp要素が2つ連続してある場合、2つのp要素のあいだのマージンは、20ピクセルではなく10ピクセルとなります(10ピクセル+10ピクセルではなく、10ピクセルと10ピクセルが重なり合って計10ピクセルとなります)。10ピクセルと20ピクセルのマージンが重なり合った場合は、20ピクセルとなります。つまり、隣接するマージンは、その中で最大のものだけが有効になるということです。このルールは、単純に隣り合う要素だけでなく、ある要素とその中に含まれる要素であっても、マージンが隣接している場合には適用されます。ただし、このようになるのは上下のマージンだけで、左右のマージンに関しては重なり合うことはありません。



上下のマージンは、隣接すると重なり合う

パディング

パディングもマージンと同様に上下左右をそれぞれ個別に設定するプロパティと、上下左右を一括して設定するプロパティが用意されています。

プロパティ名	設定対象	指定できる値の数
padding-top	上のパディング	1
padding-bottom	下のパディング	1
padding-left	左のパディング	1
padding-right	右のパディング	1
padding	上下左右のパディング	1～4

パディングを設定するプロパティ

指定できる値もマージンと同様ですが、パディングの場合は「auto」は指定できません。

パディング関連のプロパティに指定できる値

- **単位つきの実数**
  パディングを単位つきの実数(30pxなど)で指定します。

- **パーセンテージ**
  この要素を含んでいるブロックレベル要素の幅に対するパーセンテージで指定します。上下のパディングについても、幅に対するパーセンテージとなります。

上下左右のパディングを一括して指定するpaddingプロパティも、marginプロパティと同様のパターンで値を4つまで指定できます。パディングに関しては、マージンのように隣り合うものが重なり合うことはありません。

ボーダー

ボーダーを設定するプロパティは、基本的なもの(CSS2.1で定義されているもの)だけで20種類あります。まずは、どんなプロパティが用意されているのかをざっと見てください。

プロパティ名	設定対象	指定できる値の数
border-top-style	上のボーダーの線種	1
border-bottom-style	下のボーダーの線種	1
border-left-style	左のボーダーの線種	1
border-right-style	右のボーダーの線種	1
border-style	上下左右のボーダーの線種	1～4
border-top-width	上のボーダーの太さ	1
border-bottom-width	下のボーダーの太さ	1
border-left-width	左のボーダーの太さ	1
border-right-width	右のボーダーの太さ	1
border-width	上下左右のボーダーの太さ	1～4
border-top-color	上のボーダーの色	1
border-bottom-color	下のボーダーの色	1
border-left-color	左のボーダーの色	1
border-right-color	右のボーダーの色	1
border-color	上下左右のボーダーの色	1～4
border-top	上のボーダーの線種と太さと色	線種／太さ／色
border-bottom	下のボーダーの線種と太さと色	線種／太さ／色
border-left	左のボーダーの線種と太さと色	線種／太さ／色
border-right	右のボーダーの線種と太さと色	線種／太さ／色
border	上下左右のボーダーの線種と太さと色	線種／太さ／色

ボーダーを設定するプロパティ。「線種／太さ／色」は半角スペースで区切って順不同で必要な値のみ指定できるが、それらのセットをさらに1～4個指定することはできない

ボーダーもマージンやパディングと同様に上下左右に個別に設定するものと一括で設定するものが用意されています。しかし、ボーダーの場合は上下左右それぞれに「線種」と「太さ」と「色」が指定できるため、プロパティの種類が多くなっています。マージンやパディングと同様に値を1～4個指定して上下左右に適用できるタイプのほかに、上下左右を別々に指定できないけれども「線種」と「太さ」と「色」を一度に指定できるプロパティも用意されています(borderプロパティは、上下左右に対して同じ線種・太さ・色を設定します)。このタイプのプロパティは必要な値だけを半角スペースで区切って指定できますが、指定しなかった値は初期値にリセットされる点に注意してください。つまり線種・太さ・色のいずれかを省略した場合、その値は現状を維持するのではなく初期値に戻されることになります。「線種」「太さ」「色」に対して指定できる値は次の通りです。

ボーダーの線種として指定できる値

- **none**
  ボーダーを表示しません。この値を指定するとボーダーの太さも0になります。表(table要素)のボーダーの線種が競合した場合は、ほかの値が優先されます。

- **hidden**
  ボーダーを表示しません。この値を指定するとボーダーの太さも0になります。表(table要素)のボーダーの線種が競合した場合は、この値が最優先されます。

- **solid**
  ボーダーの線種を実線にします。

- **double**
  ボーダーの線種を二重線にします。

- **dotted**
  ボーダーの線種を点線にします。

- **dashed**
  ボーダーの線種を破線にします。

- **groove**
  ボーダーの線自体が溝になっているようなボーダーにします。

- **ridge**
  ボーダーの線自体が盛り上がっているようなボーダーにします。

- **inset**
  ボーダーの内側の領域全体が低く見えるようなボーダーにします。

- **outset**
  ボーダーの内側の領域全体が高く見えるようなボーダーにします。

ボーダーの太さとして指定できる値

- **単位つきの実数**
  ボーダーの太さを単位つきの実数(5pxなど)で指定します。

- **thin, medium, thick**
  「細い」「中くらい」「太い」という意味のキーワードで指定できます(実際に表示される太さはブラウザによって異なります)。

ボーダーの色として指定できる値

- **色**
  色の書式に従って任意のボーダーの色を指定します。

- **transparent**
  ボーダーの色を透明にします。

▶▶ ボーダーの線種の初期値に注意

ボーダーを指定するときに注意することは、ボーダーの線種の初期値が「none」になっている点です。線種が「none」であるということは、ボーダーが表示されないだけでなく太さも0になります。つまり、線種として「none」以外の値を指定するまでは、いくら太さや色を指定してもボーダーは表示されないのです。

▶▶ ボーダーの表示例

以下は、ボーダーの指定例と表示例です。おなじ線種と色を指定していても、ブラウザの種類によって表示結果が微妙に異なるということも覚えておきましょう。

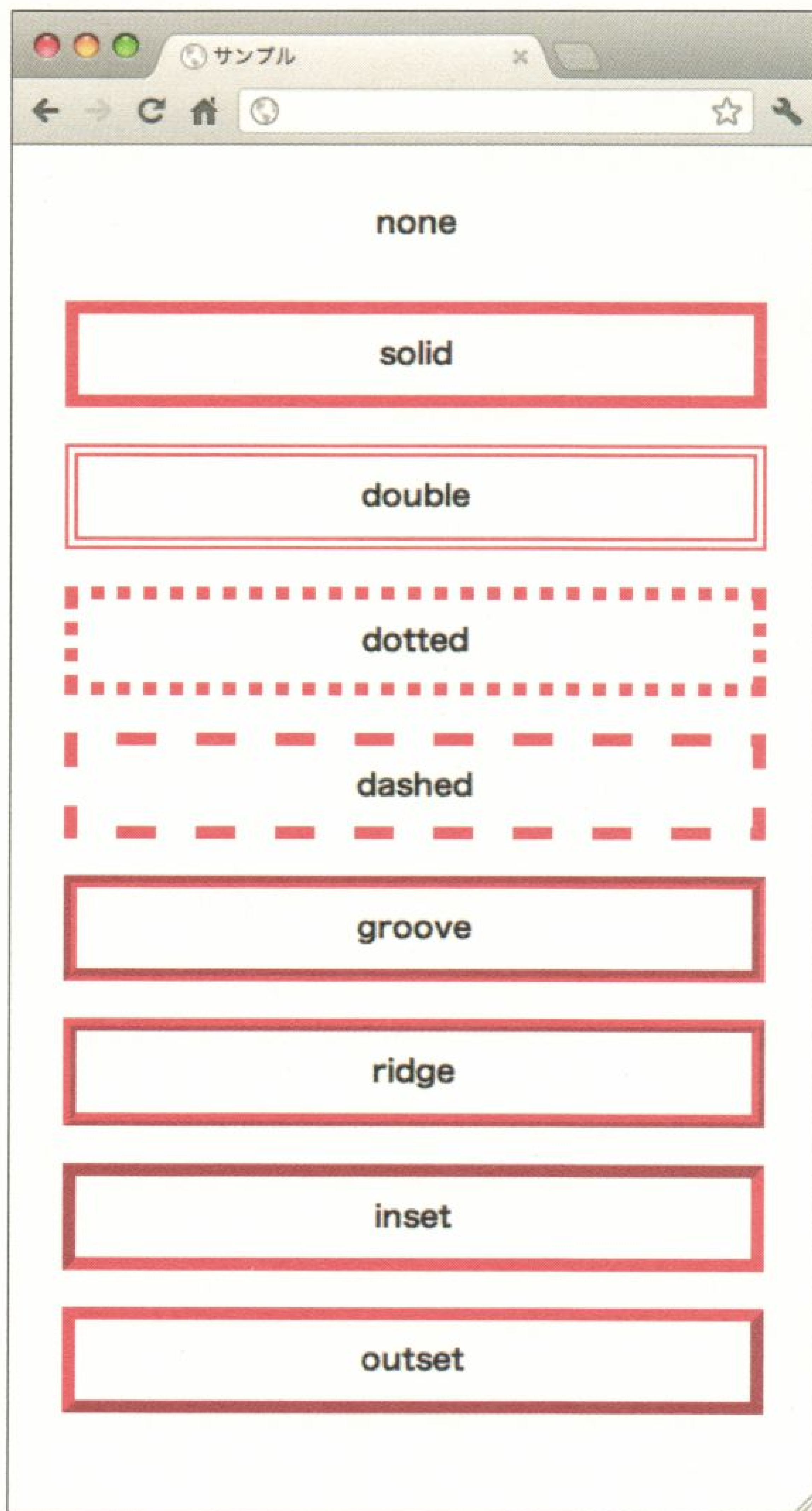
HTML sample/chapter-07/lecture-7-3/01.html

```
<p id="sample1">none</p>
<p id="sample2">solid</p>
<p id="sample3">double</p>
<p id="sample4">dotted</p>
<p id="sample5">dashed</p>
<p id="sample6">groove</p>
<p id="sample7">ridge</p>
<p id="sample8">inset</p>
<p id="sample9">outset</p>
```

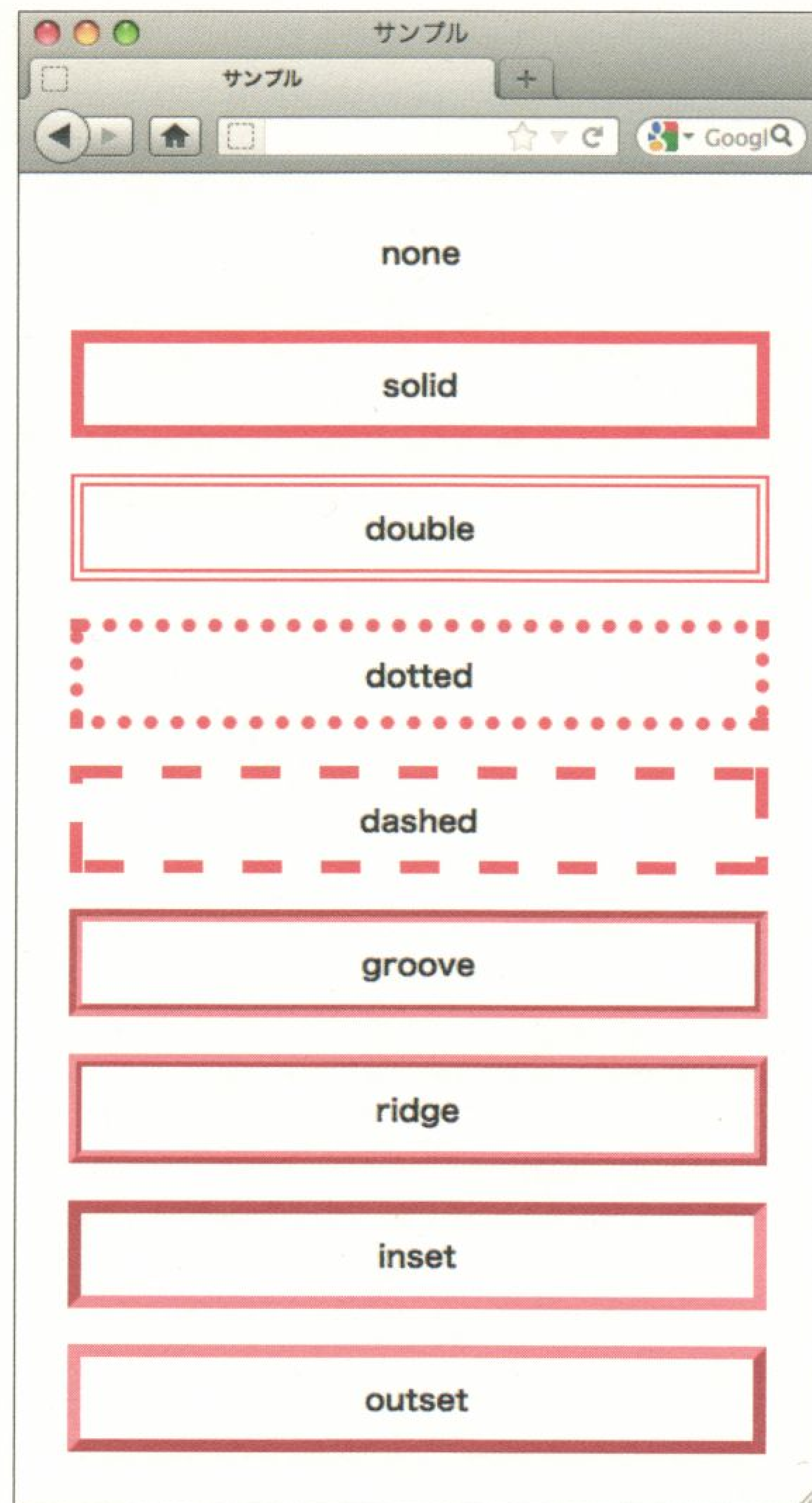
CSS sample/chapter-07/lecture-7-3/01-border.css

```
p {
  margin: 20px;
  border: solid 7px red;
  padding: 10px;
  text-align: center;
  font-weight: bold;
}
#sample1 { border-style: none; }
#sample2 { border-style: solid; }
#sample3 { border-style: double; }
#sample4 { border-style: dotted; }
#sample5 { border-style: dashed; }
#sample6 { border-style: groove; }
#sample7 { border-style: ridge; }
#sample8 { border-style: inset; }
#sample9 { border-style: outset; }
```

ボーダー関連プロパティの使用例



前ページのソースコードのGoogle Chromeでの表示例



前ページのソースコードのFirefoxでの表示例。「dotted」はGoogle Chromeでは四角の点になるのに対し、Firefoxでは丸い点になる。また、「groove」以下の立体的な表示の色あいもGoogle Chromeとは微妙に違っている

CHAPTER 7

幅と高さ

次は、ボックスの「要素内容を表示する領域」の幅と高さを設定するプロパティです。widthとheightは大きさを固定的に指定するときに使うプロパティで、それらの前に「min-」または「max-」がついたプロパティは、最小値また最大値を設定して幅や高さを制限する場合に使用します。それぞれ次のような値が指定できます。

プロパティ名	設定対象	指定できる値の数
width	幅	1
height	高さ	1
min-width	最小の幅	1
max-width	最大の幅	1
min-height	最小の高さ	1
max-height	最大の高さ	1

幅と高さを設定するプロパティ

widthに指定できる値

- **単位つきの実数**
  幅を単位つきの実数で指定します。
- **パーセンテージ**
  この要素を含んでいるブロックレベル要素の幅(widthプロパティの値)に対するパーセンテージで指定します。
- **auto**
  幅をボックスの状況から自動設定します。

heightに指定できる値

- **単位つきの実数**
  高さを単位つきの実数で指定します。
- **パーセンテージ**
  この要素を含んでいるブロックレベル要素の高さ(heightプロパティの値)に対するパーセンテージで指定します。この要素を含んでいるブロックレベル要素の高さが特に指定されていない場合は、高さは「auto」となります。
- **auto**
  高さをボックスの状況から自動設定します。

min-widthに指定できる値

・単位つきの実数
最小の幅を単位つきの実数で指定します。

・パーセンテージ
この要素を含んでいるブロックレベル要素の幅（widthプロパティの値）に対するパーセンテージで指定します。

max-widthに指定できる値

・none
幅の制限をしません。

・単位つきの実数
最大の幅を単位つきの実数で指定します。

・パーセンテージ
この要素を含んでいるブロックレベル要素の幅（widthプロパティの値）に対するパーセンテージで指定します。

min-heightに指定できる値

・単位つきの実数
最小の高さを単位つきの実数で指定します。

・パーセンテージ
この要素を含んでいるブロックレベル要素の高さ（heightプロパティの値）に対するパーセンテージで指定します。この要素を含んでいるブロックレベル要素の高さが特に指定されていない場合、値は「0」となります。

max-heightに指定できる値

・none
高さの制限をしません。

・単位つきの実数
最大の高さを単位つきの実数で指定します。

・パーセンテージ
この要素を含んでいるブロックレベル要素の高さ（heightプロパティの値）に対するパーセンテージで指定します。この要素を含んでいるブロックレベル要素の高さが特に指定されていない場合、値は「none」となります。

▶▶ パーセンテージで高さを指定する場合の注意

ここで紹介しているプロパティは、単純に大きさを指定したサイズに変更したり、大きさを制限するだけで、使用方法が特に難しいというものではありません。ただし、パーセンテージで高さを指定する際は、その要素を含んでいるブロックレベル要素の高さがきちんと指定されているかどうか注意してください。

たとえば、以下の例ではdiv要素の高さとして「100%」を指定していますが、表示結果を見ると高さは100%にはなっていません。

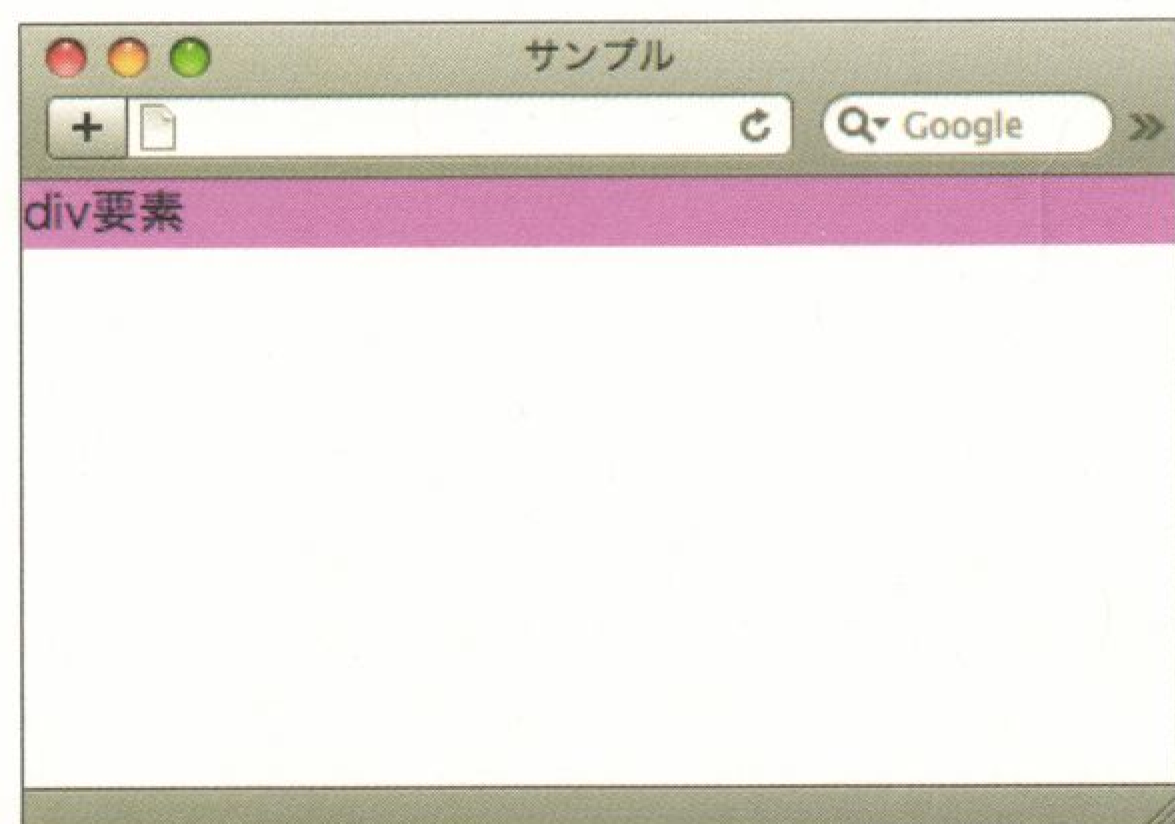
HTML

```
<div>div要素</div>
```

CSS

```
html, body {
  margin: 0;
  padding: 0;
}
div {
  height: 100%;
  background-color: #ff66ff;
}
```

div要素の高さとして「100%」を指定している例



上のソースコードのブラウザでの表示結果

これは、div要素を含んでいるbody要素の高さが指定されていないことが原因です。そのため、heightプロパティの値として「100%」を指定しているにもかかわらず、結果として値は「auto」となっているのです。しかし、body要素の高さを指定するだけでは、まだ足りません。body要素を含んでいるhtml要素の高さが指定されていないからです(html要素を含む要素はありませんので

これ以上は考える必要はありません)。結果として、以下のような指定を追加して、body要素とhtml要素の両方に「height: 100%;」の指定が適用されるようにすると、div要素の高さが100%になります。

HTML sample/chapter-07/lecture-7-3/02.html

```
<div>div要素</div>
```

CSS sample/chapter-07/lecture-7-3/02-width-height.css

```
html, body {
  margin: 0;
  padding: 0;
  height: 100%;  ← この指定も追加
}
div {
  height: 100%;
  background-color: #ff66ff;
}
```

body要素とhtml要素にも「height: 100%;」を指定する



上のソースコードのブラウザでの表示結果

COLUMN

widthとheightの発音

これらはよく使うプロパティであるにもかかわらず、会話の中で取り上げるときに何と言えば良いのか迷うプロパティでもあります。まず、「height」は英語の発音ではハイトと読みます。「width」の方はちょっとやっかいで、英語の発音では「dth」の部分に母音が含まれていないため、日本人には「dth」はまとまって「ズ」のように聞こえ、結果として「width」は「ウィズ」のように聞こえます。しかし、日本語の会話の中で「ウィズ」と言っても、「width」を連想する人は多くはありません。現時点でもっとも確実に伝わるのは、あえて「width」だけは日本語にして「幅」と言うことのようです。

角を丸くする【CSS3新】

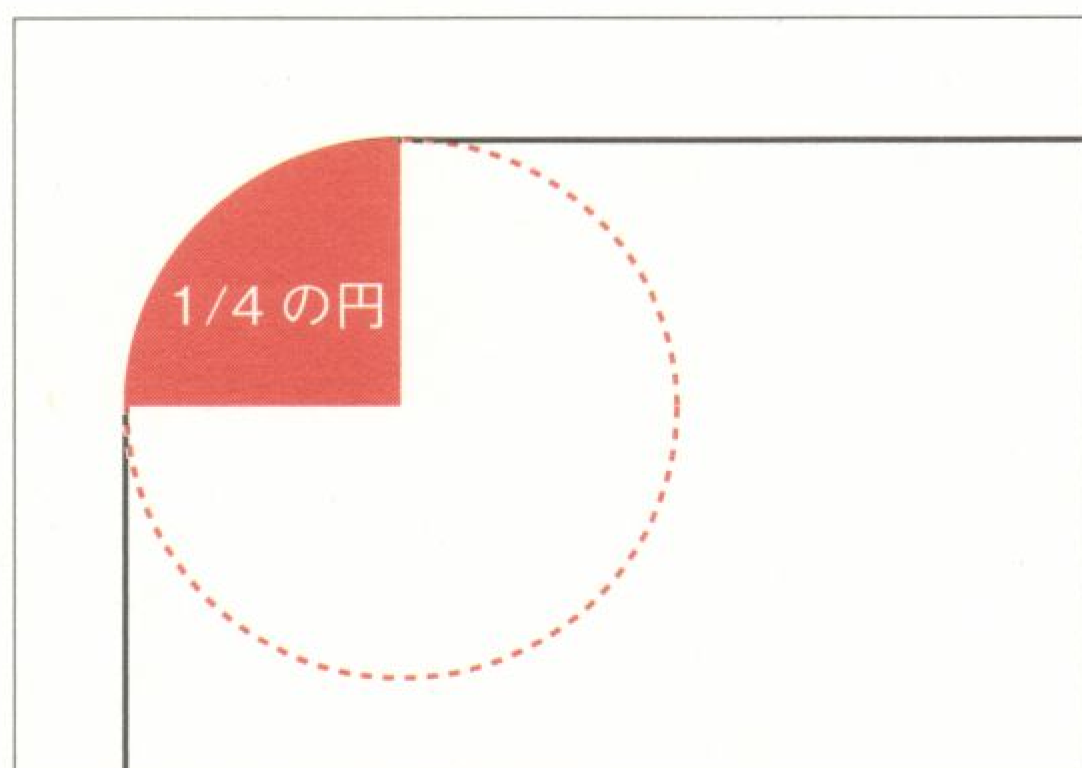
ボックスの角は、丸くすることもできます。ただし、ここまでに紹介してきたマージンやパディング、ボーダー、幅と高さなどの指定はCSS2.1でも定義されていたため基本的にどのブラウザでも使用できますが、角を丸くするプロパティに関しては勧告候補（2012年7月現在）の段階のCSS3プロパティであるため、Internet Explorer 8以前のバージョンでは未対応となっている点に注意してください（それ以外のブラウザは比較的古いバージョンから対応しています）。角を丸くするプロパティには次のものが用意されており、以下に示した値が指定できます。指定する値は、角を1/4の円にみたてたときの半径です。

プロパティ名	設定対象	指定できる値の数
border-top-left-radius	左上の角丸	1
border-top-right-radius	右上の角丸	1
border-bottom-right-radius	右下の角丸	1
border-bottom-left-radius	左下の角丸	1
border-radius	上下左右の角丸	1～4

ボックスの角を丸くするプロパティ

角丸を設定するプロパティに指定できる値

- **単位つきの実数**
  角の半径を単位つきの実数で指定します。
- **パーセンテージ**
  角の半径をボーダー領域までの大きさに対するパーセンテージで指定します。



値には、角を1/4の円とみたてたときの半径を指定する

これらはボックスの角に関する指定ですので、これまでのように「上下左右」ではありませんが、同様の4つの角に関する個別のプロパティが用意されており、また一括指定のプロパティでは次のように値を最大4つまで指定することができます。

値の数	各値の適用場所	指定例
1	4つの角すべて	border-radius: 10px;
2	左上と右下　右上と左下	border-radius: 10px 20px;
3	左上　右上と左下　右下	border-radius: 10px 20px 30px;
4	左上　右上　右下　左下	border-radius: 10px 20px 30px 40px;

border-radiusプロパティの値の指定パターン

▶▶ ブラウザの対応状況について

角を丸くするプロパティはすでに草案ではなくなっていますので、新しいブラウザだけを対象とするのであればベンダープレフィックスをつけることなく使用できます。Firefox 3.6以前やSafari 4以前にも対応させたい場合に限り、ベンダープレフィックスをつけた指定も追加してください。Internet ExplorerとOperaに関しては、草案段階を過ぎてから対応したため、ベンダープレフィックスをつけた指定には対応していません(つまり、Internet ExplorerとOpera向けのベンダープレフィックスは不要です)。

HTML　**sample/chapter-07/lecture-7-3/03.html**

```
<p>角を丸くしています</p>
```

CSS　**sample/chapter-07/lecture-7-3/03-border-radius.css**

```
p {
  padding: 3em;
  color: white;
  background-color: silver;
  -webkit-border-radius: 10px;
  -moz-border-radius: 10px;
  border-radius: 10px;
}
```

border-radiusプロパティの指定例



border-radiusを指定したボックスの表示例

COLUMN

Firefox 3.6以前に対応させる際の注意

Firefox 3.6以前であっても、border-radiusプロパティであれば「-moz-border-radius」のようにベンダープレフィックスをつけるだけで問題なく使用できます。しかし、角を個別に設定するプロパティに関しては、その名前自体も以下のように変更する必要がありますので注意してください。

CSS3で定義されている名前	Firefox 3.6以前がサポートしている名前
border-top-left-radius	-moz-border-radius-topleft
border-top-right-radius	-moz-border-radius-topright
border-bottom-right-radius	-moz-border-radius-bottomright
border-bottom-left-radius	-moz-border-radius-bottomleft

LECTURE 7-4

背景を指定する（2）

Chapter 4で背景関連のプロパティを一部紹介しましたが、背景に関連するプロパティはそれ以外にもまだまだたくさんあります。実は、背景に関する指定はボックスの各表示領域とも密接に関わりあっているため、ボックスに関する説明をしていない段階では正確な仕様を伝えることが難しいプロパティが多かったのです。というわけで、ここではボックスの構造をふまえた上で、背景関連の残りのプロパティを紹介していきます。

背景画像の表示位置を指定する

background-position プロパティを使用すると、背景画像の表示位置を指定することができます。background-repeatプロパティの値が「no-repeat」の場合（背景画像を1つだけ表示させる場合）は背景画像がその位置に表示されますが、背景画像を繰り返して表示させる場合にはその位置を基準にして縦または横に繰り返されることになります。
指定できる値は次の通りです。

background-positionに指定できる値

- **単位つきの実数**
  表示位置を単位つきの実数で指定します。
- **パーセンテージ**
  表示位置をパーセンテージで指定します。
- **top, bottom, left, right, center**
  表示位置をキーワードで指定します。topは縦方向の「0%」、bottomは「100%」と同じです。同様に、leftは横方向の「0%」、rightは「100%」と同じです。centerは縦方向または横方向の「50%」と同じです。

表示位置は、横方向・縦方向の順に半角スペースで区切って2つ指定します（ただし、値をキーワードのみで指定するのであれば順番は逆でもかまいません）。値を1つしか指定しなかった場合は、もう一方に「center」が指定されたものとして処理されます。

CHAPTER 7

表示位置の基準

表示位置の基準は、パディング領域の左上です。パディング領域の左上に背景画像の左上がぴったり重なった状態が「0px 0px」「0% 0%」「left top」です。逆に、パディング領域の右下に背景画像の右下がぴったり重なった状態は「100% 100%」「right bottom」となります。背景画像がパディング領域の中央にある状態は「50% 50%」「center center」です。

値を「単位つきの実数」で指定した場合、横方向はパディング領域の左から背景画像の左までの距離、縦方向はパディング領域の上から背景画像の上までの距離となります。値を「パーセンテージ」で指定した場合は、横方向・縦方向ともにパディング領域のその％のポイントと背景画像のその％のポイントを重ねた状態となる位置に表示されます。値を数値で指定する場合には、マイナスの値も指定できます。

背景画像の表示例

ではここで、背景画像をページ全体の中央に表示させる例を紹介しておきましょう。ソースコードは次の通りです。

HTML

```
・・・
<body>

</body>
</html>
```

CSS

```
html, body {
  margin: 0;
}
body {
  background-image: url(images/shell.jpg);
  background-repeat: no-repeat;
  background-position: center;
}
```

背景画像をページ全体の中央に表示させる例

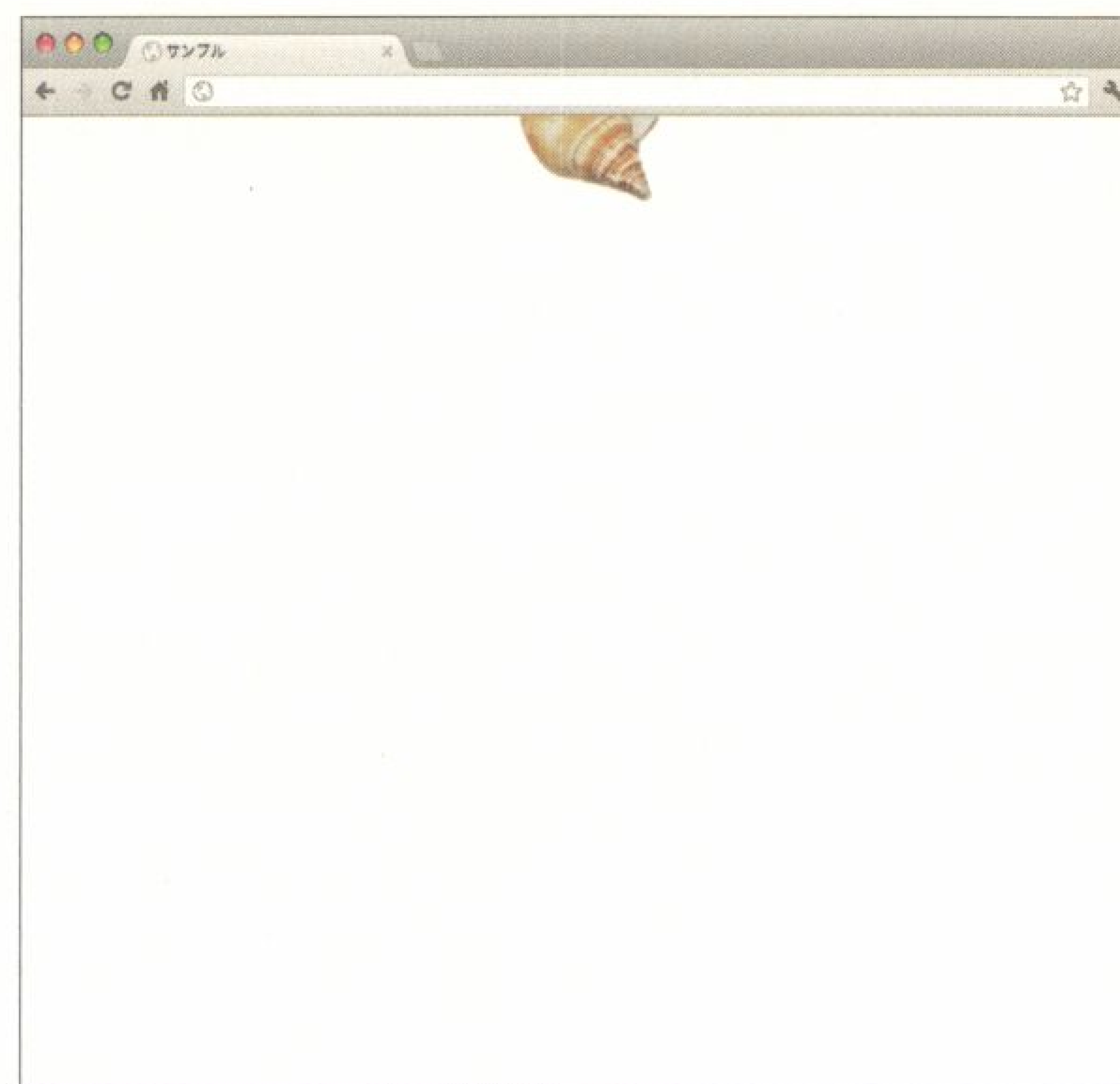


背景画像として指定している画像「shell.jpg」

しかし、実際にこのソースコードをブラウザで表示させてみると、表示結果は次ページのようになります。

背景画像は横方向の中央には配置されているのですが、縦方向は中央どころかマイナスの数値を指定したような状態となっています。なぜこのような表示結果となってしまったのでしょうか？　background-positionプロパティの値には「center」だけを指定していますので、もう一方も「center」になり、「center center」を指定した場合と同じ表示結果になるはずです。

実は、この背景画像はまったく指定通りに表示されていると言えます。背景画像は指定通りに、body要素の縦横の中央にしっかりと表示されているのです。では、何がおかしいのでしょうか？


前ページのソースコードの表示例

実はbody要素の内容が空なので、body要素の高さが0となっているのです。高さが0のボックスの中央に合わせて表示されているので、上のような表示結果となったわけです。

したがって、body要素の高さを「100%」にすればいいのですが、高さをパーセンテージで指定する場合には、それを含む要素にも高さを指定する必要があります。結果として、次の指定を追加することで、背景画像はページの中央に表示されるようになります。

HTML　　**sample/chapter-07/lecture-7-4/01.html**

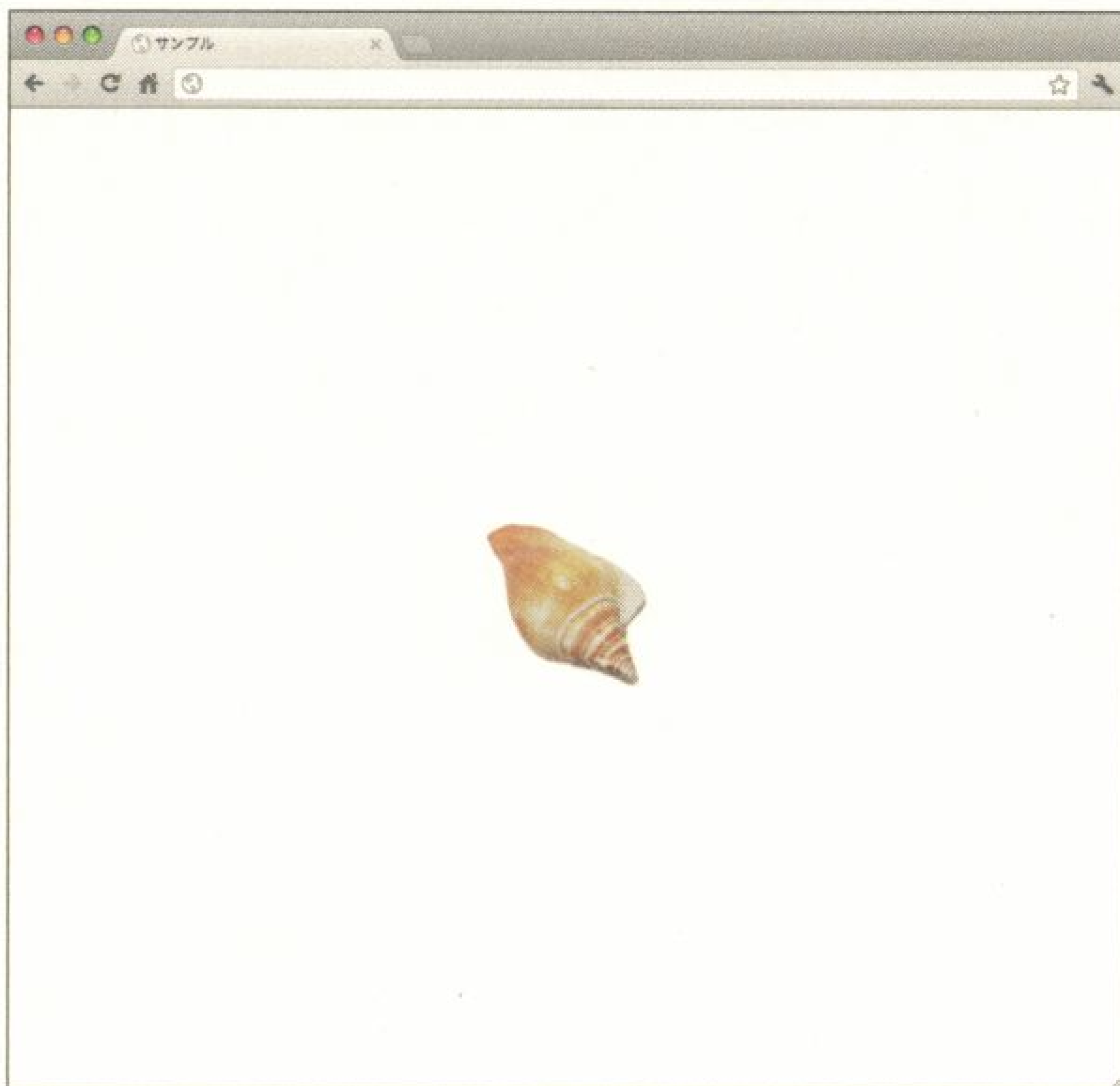
```
・・・
<body>

</body>
</html>
```

CSS　　**sample/chapter-07/lecture-7-4/01-bg-position.css**

```
html, body {
  margin: 0;
  height: 100%;  ← この指定を追加
}
body {
  background-image: url(images/shell.jpg);
  background-repeat: no-repeat;
  background-position: center;
}
```

先程のCSSのソースコードに、body要素とhtml要素の高さの指定も追加



背景画像がページの中央に表示されるようになった

背景画像をウィンドウに固定する

長いページをスクロールすると、通常はコンテンツと一緒に背景画像もスクロールします。background-attachmentプロパティの値として「fixed」を指定すると、背景画像はウィンドウ上に固定され、スクロールしても動かなくなります。

background-attachmentに指定できる値

- scroll
  背景画像は他のコンテンツと一緒にスクロールします。

- fixed
  背景画像をウィンドウ上に固定して、スクロールしても動かないようにします。

なお、background-attachmentプロパティの値として「fixed」を指定すると、background-positionプロパティでの配置の基準が「パディング領域」から「ウィンドウの表示領域（見えている領域）全体」となりますので注意してください。

▶▶ 背景画像の固定の表示例

以下は、背景画像を繰り返しなしで右下に配置して、background-attachmentプロパティの値を「fixed」にした例です。スクロールさせても背景画像の位置は変わりません。

HTML　　sample/chapter-07/lecture-7-4/02.html

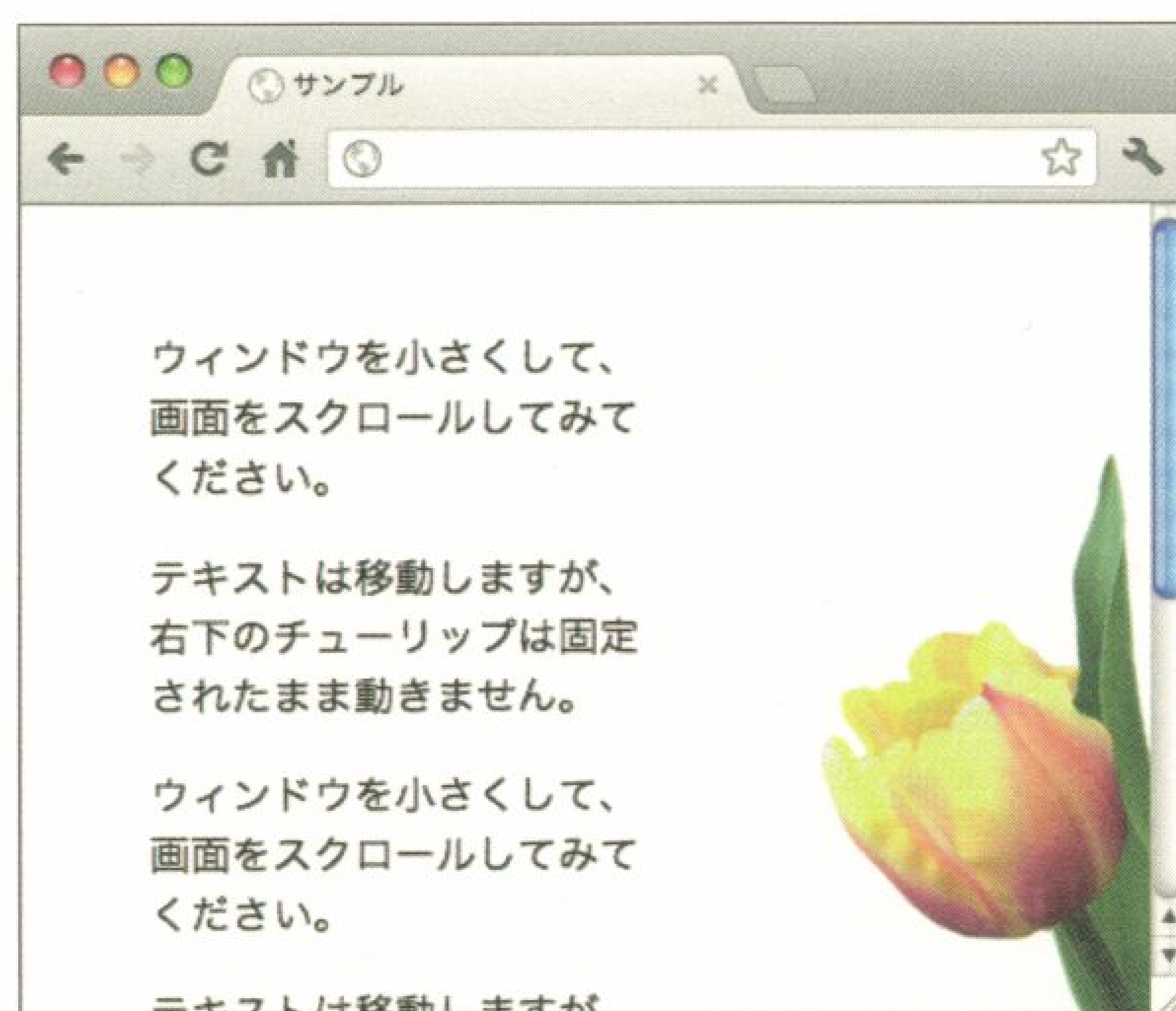
```
<p>
ウィンドウを小さくして、画面をスクロールしてみてください。
</p>
<p>
テキストは移動しますが、右下のチューリップは固定されたまま動きません。
</p>

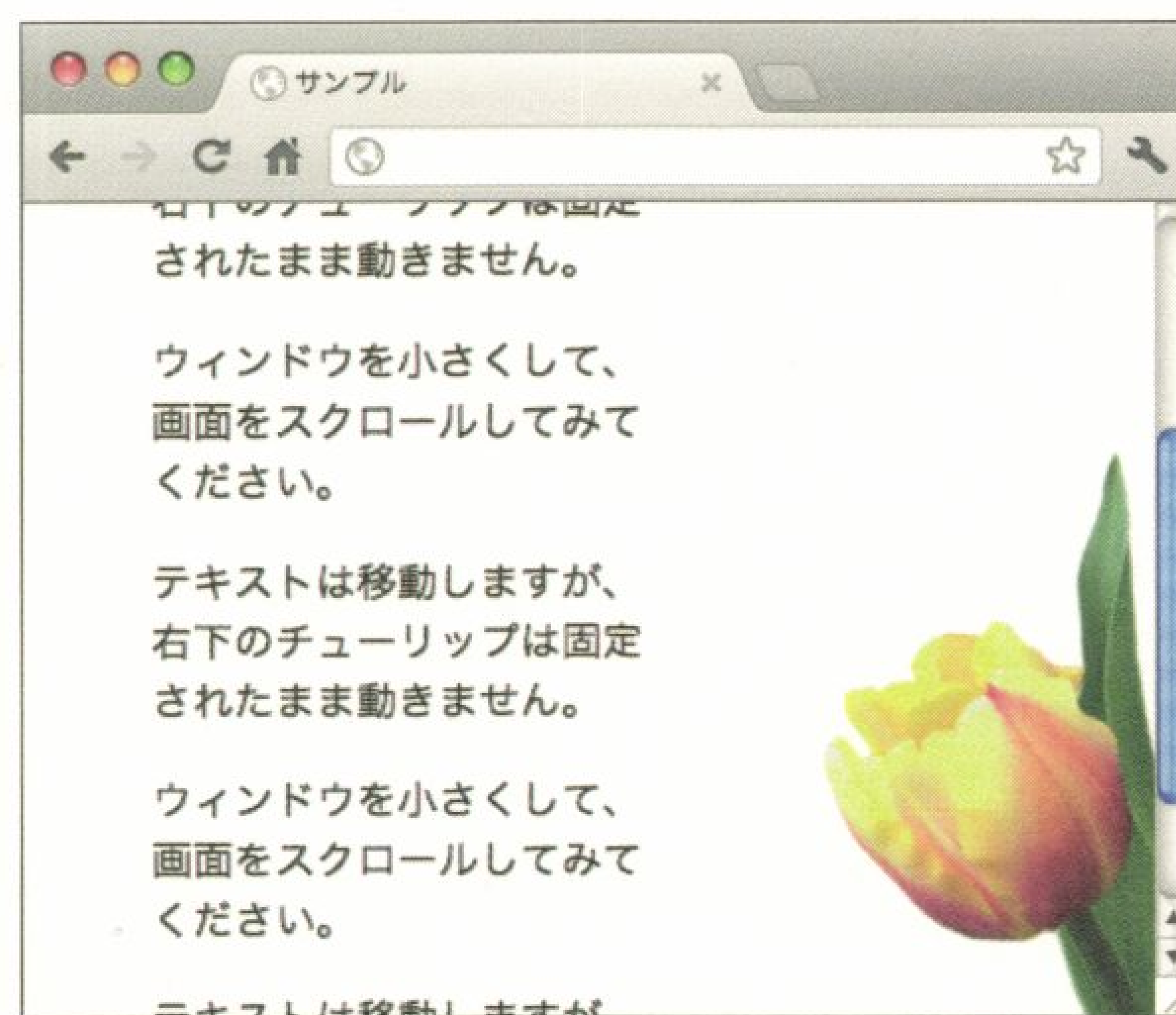
・・・以下、同じテキストの繰り返し・・・
```

CSS　　sample/chapter-07/lecture-7-4/02-bg-attachment.css

```
body {
  margin: 50px 200px 0 50px;
  background-image: url(images/tulip.png);
  background-repeat: no-repeat;
  background-position: right bottom;
  background-attachment: fixed;
}
```

background-attachmentプロパティの指定例





上のソースコードの表示例。スクロールさせても背景画像の位置は変わらない

COLUMN

数値が0のときは単位を省略できる

さて、すでにお気付きの方もいらっしゃるかもしれませんが、先程のサンプルのmarginプロパティの値の中に、単位のついていない値が混じっていました。

```
body {
  margin: 50px 200px 0 50px;
  ・・・
}
```

「0」に単位がついていない

実は、値が「単位つきの実数」であっても、その値が0である場合には単位は省略できることになっているのです。今後のサンプルでも、値が0の場合には基本的に0を省略しますので、覚えておいてください。

背景画像の表示サイズを変更する 【CSS3新】

background-sizeは背景画像の表示サイズを指定するプロパティです。次の値が指定できます。

background-sizeに指定できる値

- **単位つきの実数**
  背景画像の幅または高さを単位つきの実数で指定します。
- **パーセンテージ**
  背景画像の幅または高さをパディング領域に対するパーセンテージで指定します。
- **auto**
  幅と高さの両方が「auto」の場合は、元のサイズで表示します。幅または高さの一方にこの値を指定すると、背景画像の縦横の比率を保ったままもう一方に合わせたサイズになります。
- **cover**
  縦横の比率を保ったまま、背景画像1つで表示領域全体を覆うサイズにします。
- **contain**
  縦横の比率を保ったまま、背景画像の全体が表示されるサイズにします。

「cover」と「contain」は単独で指定しますが、それ以外の値については半角スペースで区切

って幅・高さの順に指定します。2つめの値を省略すると、高さに「auto」を指定した状態となります。

▶▶ ブラウザの対応状況について

このプロパティはCSS3で新しく定義されたもので、Internet Explorer 8以前は対応していません。Firefox やSafariの少し古いバージョンにも対応させたい場合は、ベンダープレフィックスつきの指定も加えてください。Internet ExplorerとOperaについては、ベンダープレフィックスつきの値はサポートしてませんので不要です。

HTML sample/chapter-07/lecture-7-4/03.html

```
・・・
<body>

</body>
</html>
```

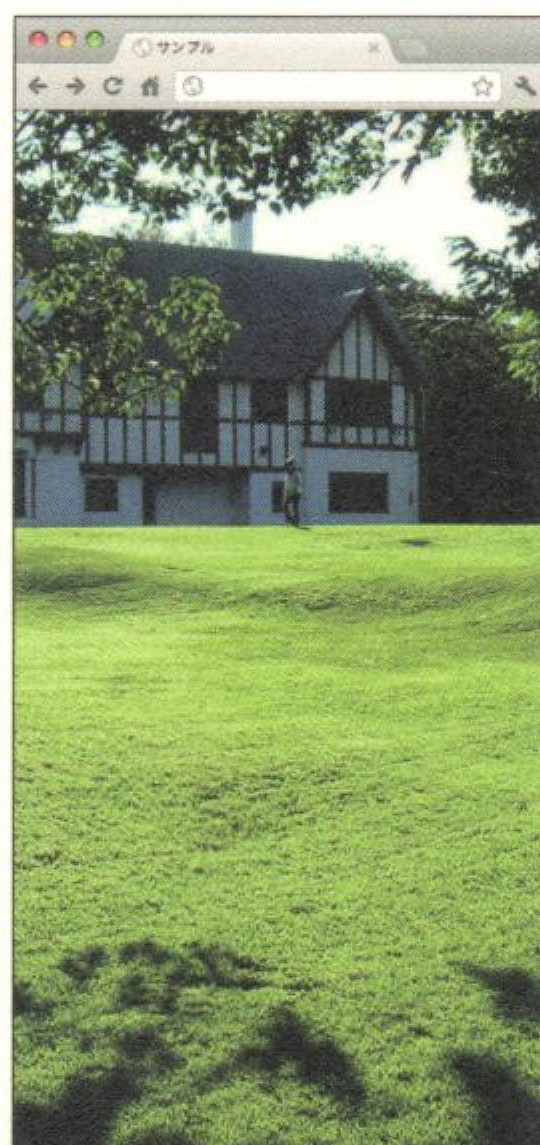
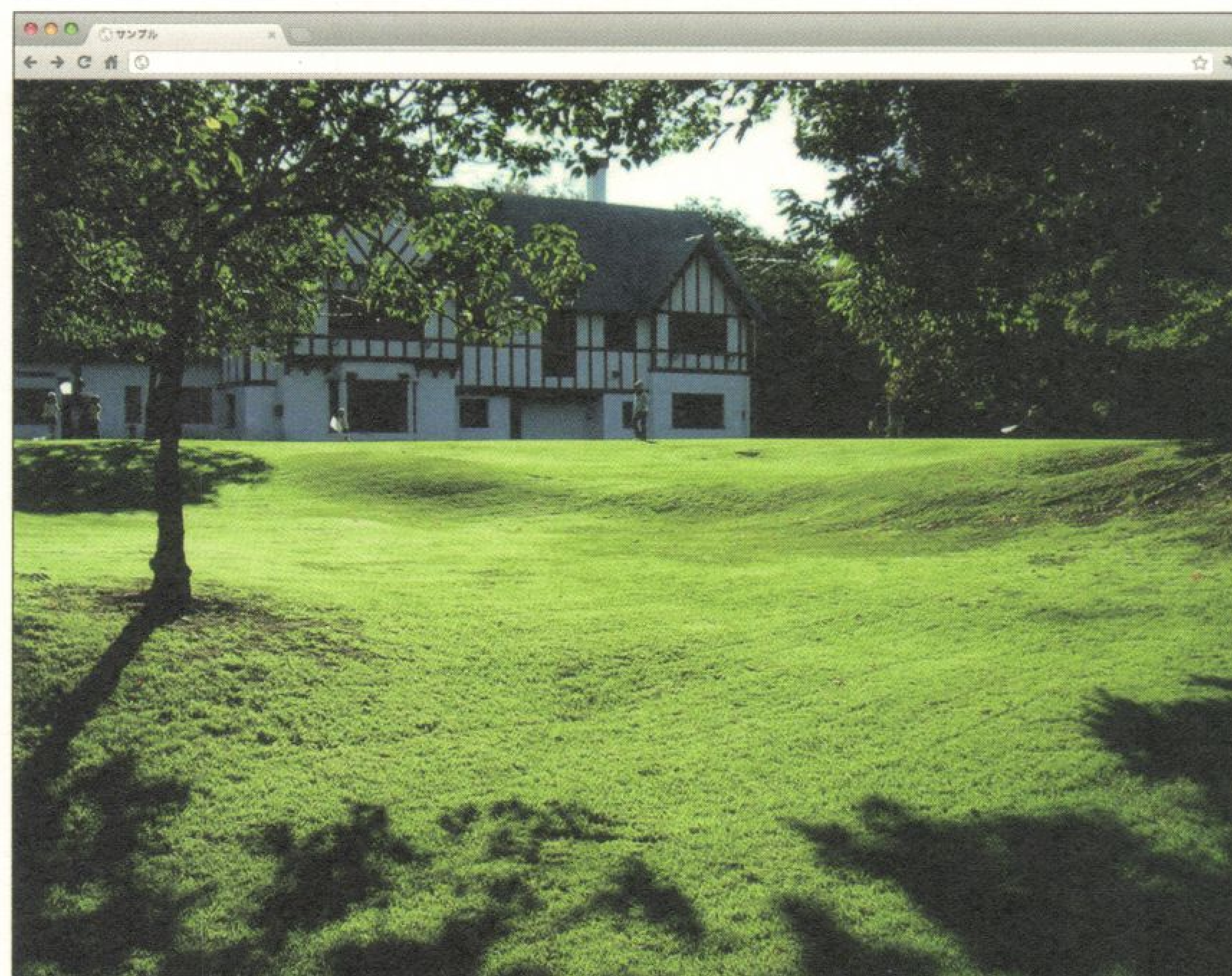
CSS sample/chapter-07/lecture-7-4/03-bg-size-cover.css

```
html, body { height: 100%; }
body {
  margin: 0;
  background-image: url(images/green.jpg);
  background-position: center;
  -webkit-background-size: cover;
  -moz-background-size: cover;
  background-size: cover;
}
```

「cover」の指定例



表示させている背景画像「green.jpg」

ウィンドウのサイズをいろいろと変更してみても、常に1つの背景画像で全体を覆っている

HTML **sample/chapter-07/lecture-7-4/04.html**

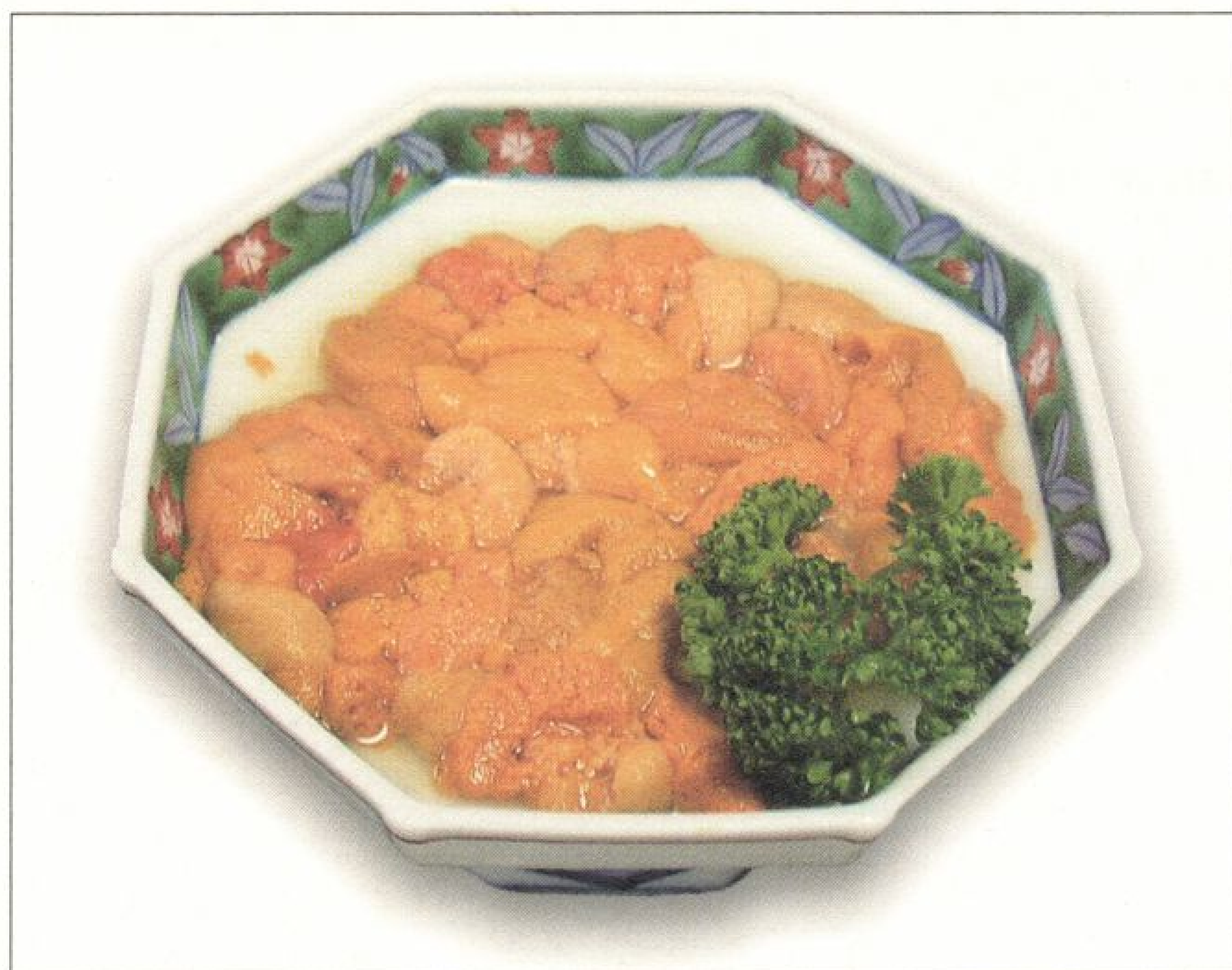
```
・・・
<body>

</body>
</html>
```

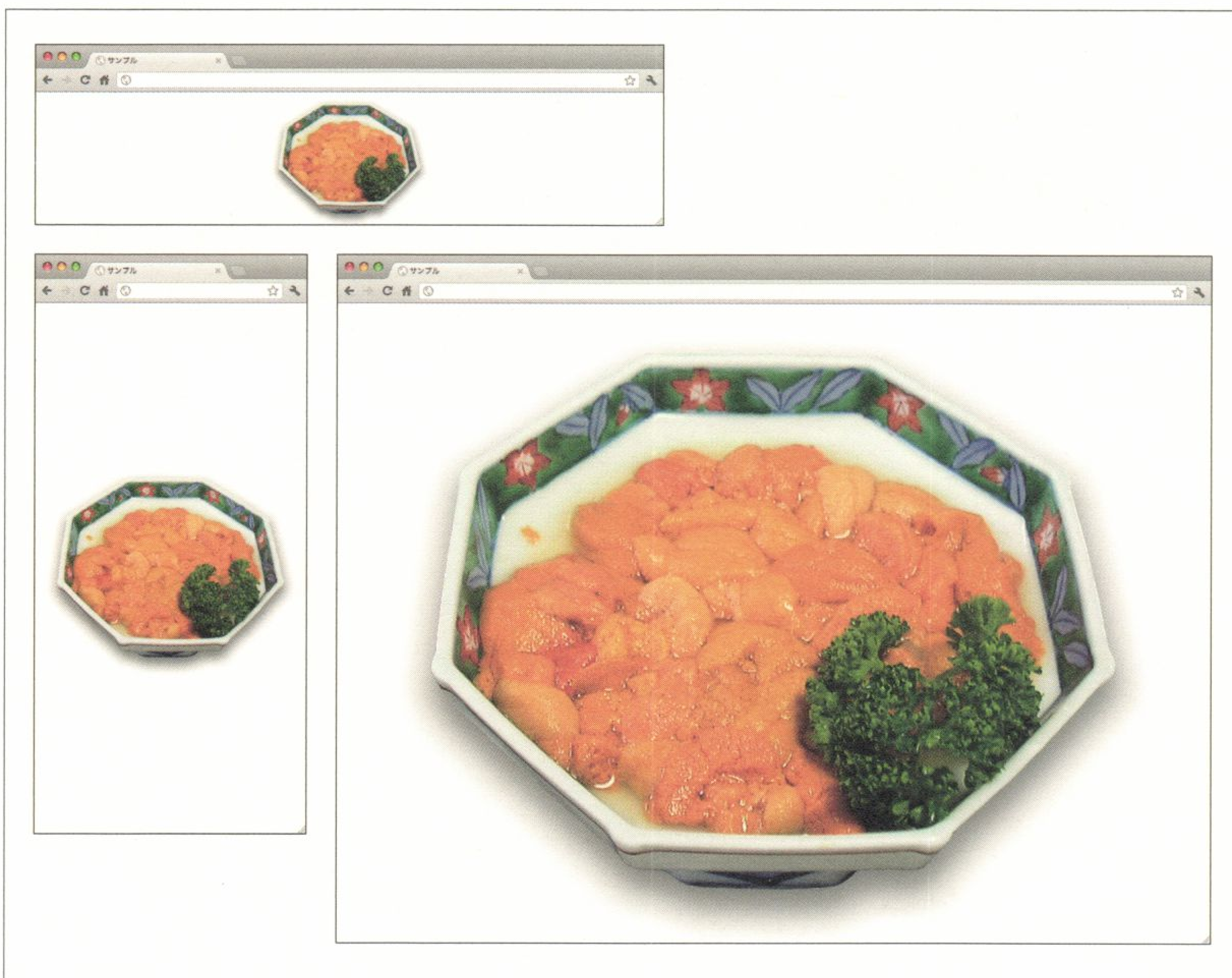
CSS **sample/chapter-07/lecture-7-4/04-bg-size-contain.css**

```
html, body { height: 100%; }
body {
  margin: 0;
  background-image: url(images/uni.jpg);
  background-position: center;
  background-repeat: no-repeat;
  -webkit-background-size: contain;
  -moz-background-size: contain;
  background-size: contain;
}
```

値「contain」の指定例



表示させている背景画像「uni.jpg」



ウィンドウのサイズをいろいろと変更してみても、常に背景画像の全体が表示されている

複数の背景画像を指定する 【CSS3改】

CSS2.1の仕様では、1つのボックスに対して1つの背景画像しか指定できないことになっています。しかし、CSS3からは1つのボックスにいくつでも背景画像を指定できます。ただし、Google ChromeやSafari、Firefoxは少々古いバージョンでも複数の背景画像に対応していますが、Internet Explorerについてはバージョン8以前は未対応である点に注意してください。

複数の背景画像を指定する方法は簡単で、単純に値をカンマで区切って複数指定するだけです。より前に(左側)に指定している背景画像ほど上に重なって表示されます。各背景画像の表示位置や繰り返しなども指定できるように、背景関連のその他のプロパティも同様にカンマで区切って複数の値を指定できるようになっています。
次のサンプルでは、body要素に3つの背景画像を指定しています。

HTML　sample/chapter-07/lecture-7-4/05.html

```
・・・
<body>

</body>
</html>
```

CSS　sample/chapter-07/lecture-7-4/05-multiple-images.css

```
html, body { height: 100%; }
body {
  margin: 0;
  background-image: url(images/wall.png), url(images/plane.png),
url(images/sky.jpg);
  background-position: bottom, 50% 10%, center;
  background-repeat: no-repeat;
  -webkit-background-size: 100% auto, auto, cover;
  -moz-background-size: 100% auto, auto, cover;
  background-size: 100% auto, auto, cover;
}
```

複数の背景画像を指定する例。背景関連のプロパティには、カンマで区切れば複数の値が指定できる


表示させている背景画像「wall.png」


表示させている背景画像「plane.png」



表示させている背景画像「sky.jpg」



各背景画像の表示位置や表示サイズを変えてあるので、ウィンドウの大きさを変えると背景もこのように変化する

背景関連プロパティの一括指定

フォント関連のプロパティの値をまとめて指定できるfontプロパティのように、背景関連のプロパティの値をまとめて一度に指定できるプロパティがあります。それが**backgroundプロパティ**です（実はChapter 2で背景色を指定したのはこのプロパティでした）。本書で紹介した次のプロパティの値はすべて指定できます。

backgroundに指定できる値

- **background-colorの値**
  background-colorに指定できる値が指定できます。

- **background-imageの値**
  background-imageに指定できる値が指定できます。

- **background-repeatの値**
  background-repeatに指定できる値が指定できます。

- **background-positionの値**
  background-positionに指定できる値が指定できます。

- **background-attachmentの値**
  background-attachmentに指定できる値が指定できます。

- **background-sizeの値**
  background-sizeに指定できる値が指定できます。

基本的には、必要な値を順不同で半角スペースで区切って指定するだけでOKです。ただし、ここで指定していない値については、現状を維持するのではなく**初期値にリセットされます**ので注意してください。

HTML sample/chapter-07/lecture-7-4/06.html

```html
・・・
<body>

</body>
</html>
```

CSS sample/chapter-07/lecture-7-4/06-background-1.css

```css
html, body { height: 100%; }
body {
  margin: 0;
    background: white url(images/shell.jpg) no-repeat center;
}
```

backgroundプロパティには背景関連のプロパティの値を順不同でまとめて指定できる

▶▶ 指定値の単位について

ただし、background-positionとbackground-sizeにはどちらも「単位つきの実数」と「パーセンテージ」が指定できるため、それらを区別できるようにしておかねければなりません。そのため、これらを指定する際には、次のようなルールがあります。

まず、「単位つきの実数」または「パーセンテージ」が1つだけ指定されている場合はbackground-positionの値だと解釈されます。そして、background-sizeの値を指定する際は、はじめにbackground-positionの値を書き、その値のあとに半角のスラッシュ(/)で区切ってbackground-sizeの値を書きます。fontプロパティの行間の値を指定するときに、font-sizeの値とのあいだをスラッシュで区切った方法と同じです。

ただし現時点では、backgroundプロパティに対してbackground-sizeの値を指定しても未対応のブラウザが多いようです。また、backgroundプロパティを使用するとベンダープレフィックスをつけた指定も組み込めなくなります。そのような理由から、現時点ではbackground-sizeプロパティの値については、backgroundプロパティで指定せずに別途指定した方が安全です。

なお、backgroundプロパティも複数の背景画像に対応しています。他の背景関連のプロパティと同様に、カンマで区切って指定できます。ただし、その場合でもbackground-colorの値は1つしか有効にできませんので、必ず一番最後(一番右側)の値として含めるようにしてください。

HTML　sample/chapter-07/lecture-7-4/07.html

```
・・・
<body>

</body>
</html>
```

CSS　sample/chapter-07/lecture-7-4/07-background-2.css

```
html, body { height: 100%; }
body {
  margin: 0;
  background: url(images/wall.png) no-repeat bottom, url(images/plane.
png) no-repeat 50% 10%, #00bfff url(images/sky.jpg) no-repeat center;
  -webkit-background-size: 100% auto, auto, cover;
  -moz-background-size: 100% auto, auto, cover;
  background-size: 100% auto, auto, cover;
}
```

backgroundプロパティに複数の背景画像を指定した例。background-sizeプロパティの値については、現時点では別に指定した方がよい

LECTURE 7-5

配置方法を指定するプロパティ

本章では重要な要素やプロパティがどんどん登場していますが、この章の最後に、現在のCSSでの配置やレイアウトの要とも言えるフロートと、その他の配置関連プロパティについて説明しておきます。

フロートの基本

フロートとは、**floatプロパティ**によって、ある要素を左または右に寄せて配置し、その反対側に後続の要素が回り込むようにした状態のことを言います。floatプロパティはインライン要素を含むほとんどの要素に指定でき、フロートの状態になるとインライン要素であってもそのボックスはブロックレベルのボックスになります。指定できる値は次の通りです。

floatに指定できる値

- `left`
  左側にフロートさせます。
- `right`
  右側にフロートさせます。
- `none`
  フロートしていない通常の状態にします。

このプロパティのごく基本的な使い方として、画像の横にテキストを回り込ませる例を示します。

HTML sample/chapter-07/lecture-7-5/01.html

```
<p id="p1">
<img src="images/photo.jpg" alt="サンプル画像">
1つめの段落のテキストです。このテキストが画像の横に回り込みます。このテキストが画像の横に回り込みます。
</p>

```

```
<p id="p2">
２つめの段落のテキストです。このテキストが画像の横に回り込みます。このテキストが画像の横に回り込みます。
</p>

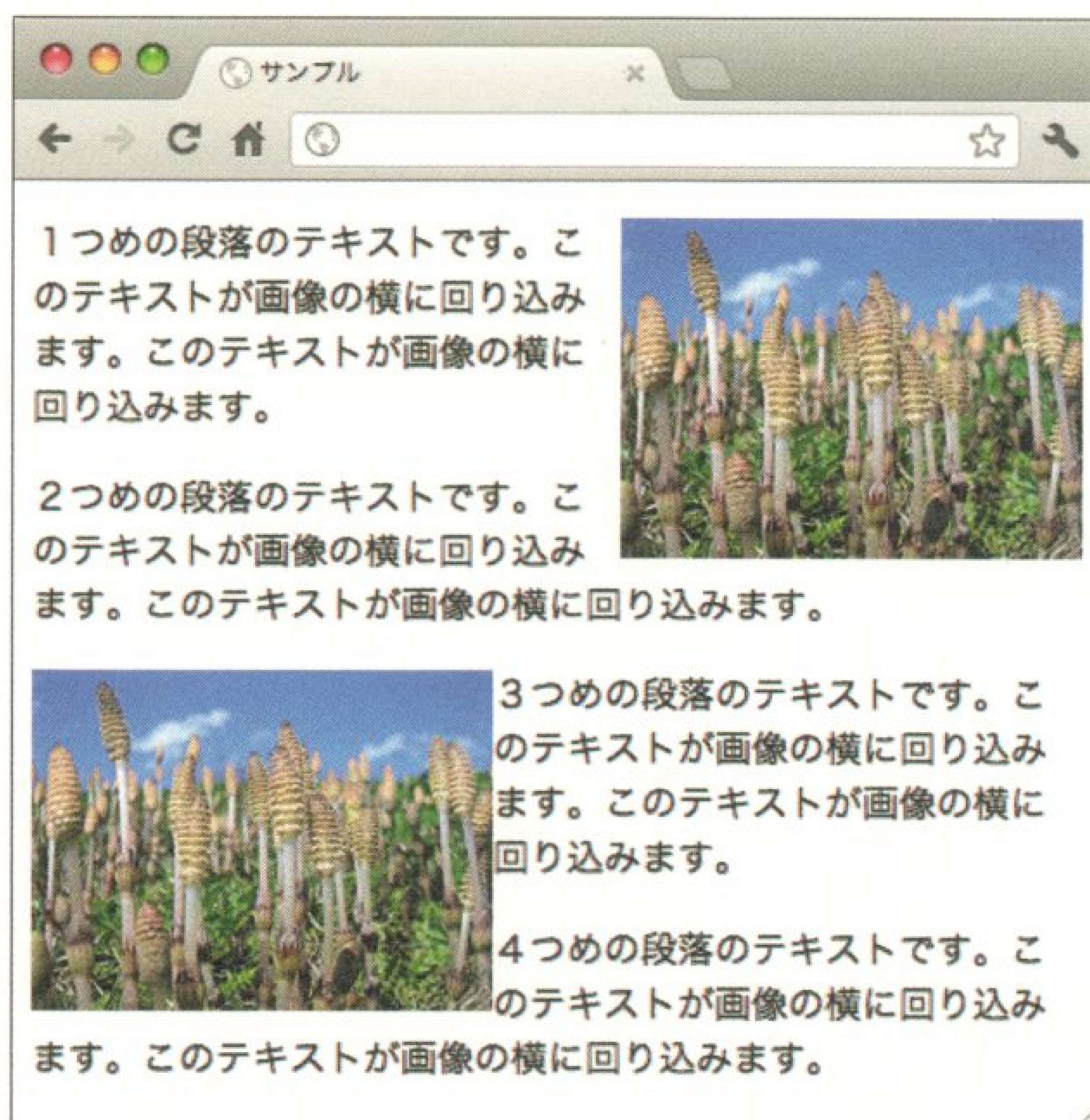
<p id="p3">
<img src="images/photo.jpg" alt="サンプル画像">
３つめの段落のテキストです。このテキストが画像の横に回り込みます。このテキストが画像の横に回り込みます。
</p>

<p id="p4">
４つめの段落のテキストです。このテキストが画像の横に回り込みます。このテキストが画像の横に回り込みます。
</p>
```

CSS　sample/chapter-07/lecture-7-5/01-float.css

```
#p1 img { float: right; }
#p3 img { float: left; }
```

1つめのimg要素を右、2つめのimg要素を左にフロートさせている例

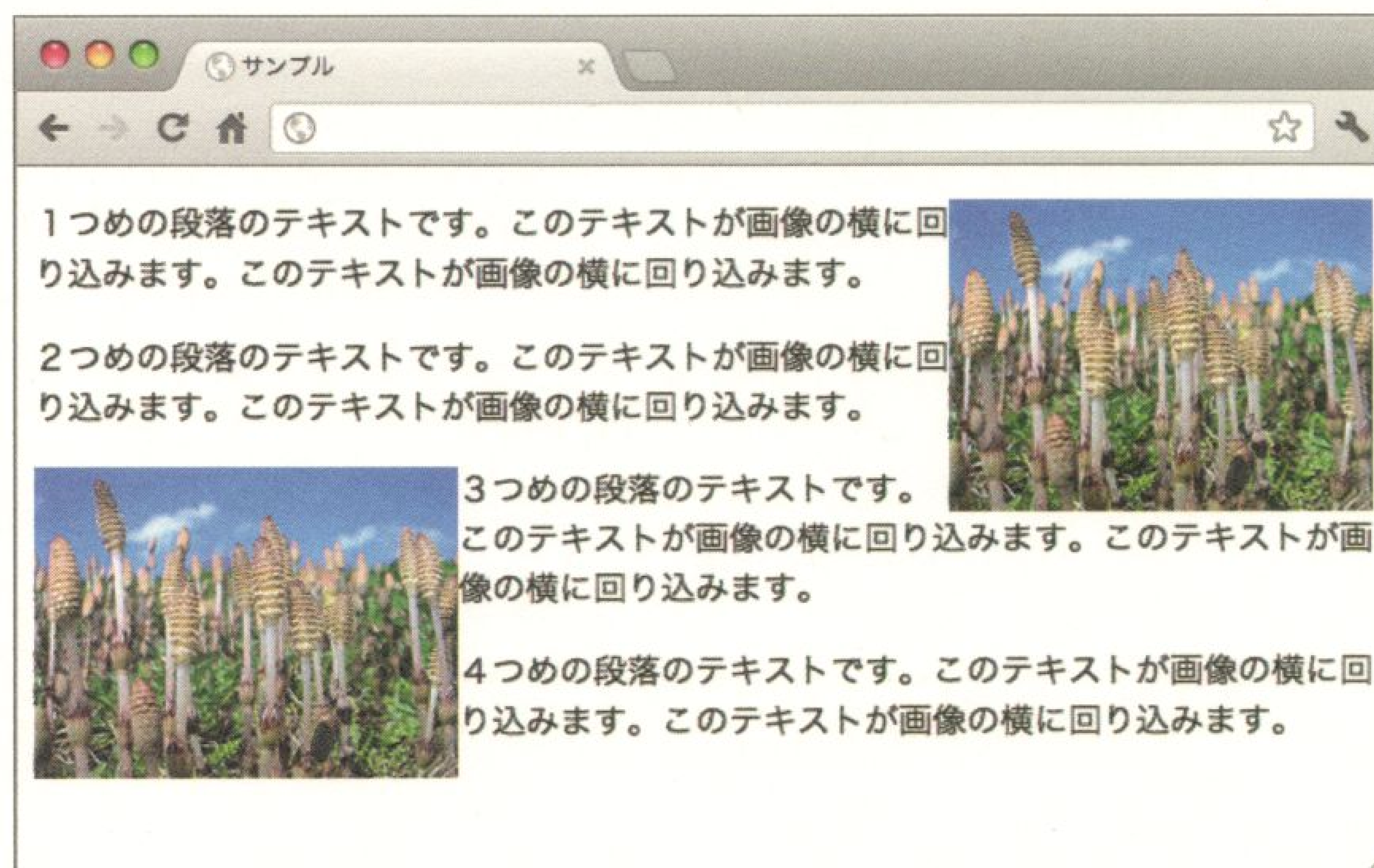


画像はそれぞれ右と左に寄せて配置され、その横にテキストが回り込んでいる

CHAPTER 7

▶▶ フロートの解除

しかし、ウィンドウの幅を少し広くしてみると、下の図のように最初の2つの段落の行数が減り、2つめの画像の位置が右の写真の下部を越えてどんどん上にあがってきてしまいます。



ウィンドウの幅を広くすると、下の画像がどんどん上にあがってくる

状況によってはこれでも問題ないかもしれませんが、2つめの画像と3段落目のテキストは常に1つめの画像よりも下に表示させたい場合もあるでしょう。そのような場合に使用するのが、指定した要素の直前でフロートを解除する**clearプロパティ**です。次の値が指定できます。

clearに指定できる値

- **left**
  この要素よりも前で「float: left;」が指定されているフロートを、この要素の直前で解除します。

- **right**
  この要素よりも前で「float: right;」が指定されているフロートを、この要素の直前で解除します。

- **both**
  この要素よりも前で指定されている左右両側のフロートを、この要素の直前で解除します。

- **none**
  フロートに解除せずにそのままにします。

clearプロパティは、**ブロックレベル要素にしか指定できない点**に注意してください。次のサンプルは、先程のサンプルの3段落目にclearプロパティを指定して右側のフロートを解除している例です。

HTML　sample/chapter-07/lecture-7-5/02-clear.css

```
<p id="p1">
<img src="images/photo.jpg" alt="サンプル画像">
１つめの段落のテキストです。このテキストが画像の横に回り込みます。このテキストが画像の横
に回り込みます。
</p>

<p id="p2">
２つめの段落のテキストです。このテキストが画像の横に回り込みます。このテキストが画像の横
に回り込みます。
</p>

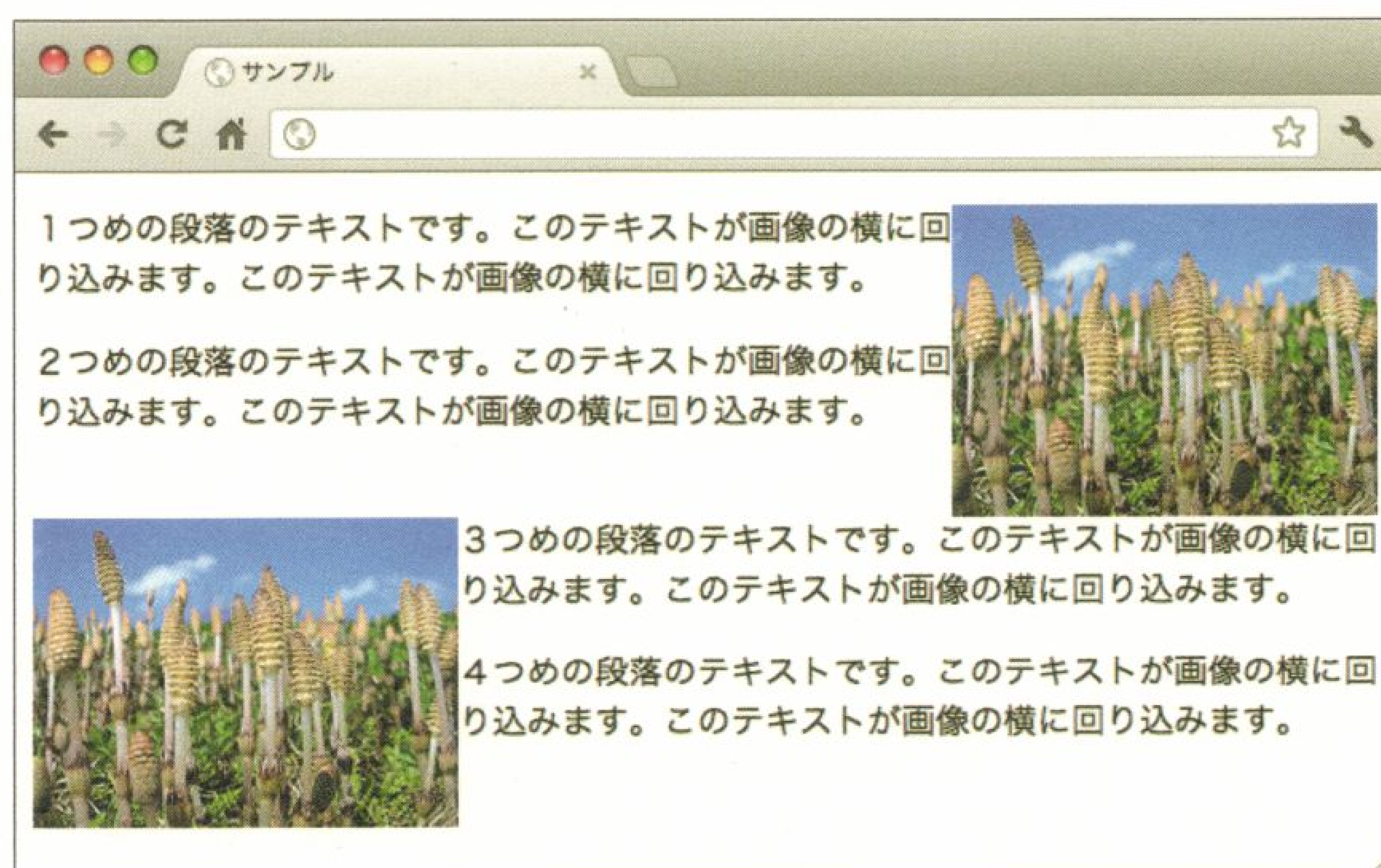
<p id="p3">
<img src="images/photo.jpg" alt="サンプル画像">
３つめの段落のテキストです。このテキストが画像の横に回り込みます。このテキストが画像の横
に回り込みます。
</p>

<p id="p4">
４つめの段落のテキストです。このテキストが画像の横に回り込みます。このテキストが画像の横
に回り込みます。
</p>
```

CSS　sample/chapter-07/lecture-7-5/02-clear.css

```
#p1 img { float: right; }
#p3 img { float: left; }
#p3 { clear: right; }
```

3つめのp要素にclearプロパティを指定



3つめのp要素の直前でフロートが解除され、3段落目のテキストと左の画像は右の画像の下から表示されている。このサンプルの場合は値にrightではなくbothを指定しても同じ表示になる

フロートによる2段組みレイアウト

CSSで段組みレイアウト(マルチカラム・レイアウト)を実現する方法はいくつかありますが、現在では多くの場合フロートが利用されています。CSS3の仕様の中には、段組みレイアウト専用の機能(フレキシブルボックス)もあるのですが、これまでに何度も仕様が変更されており、ブラウザ側の実装もまだ安定した状態にはなっていないため、現時点ではまだ利用しない方が無難です。

▶▶ 段にする範囲をグループ化する

フロートで段組みをする方法にも色々あるのですが、ここではまずシンプルで広く利用されている方法を紹介します。簡単に言えば、各段にする範囲をdiv要素でグループ化し、そのdiv要素を左右いずれかにフロートさせるだけです。このとき、通常はすべての段とそれらを含むdiv要素にも幅を指定します。それでは簡単な2段組みの例から見てみましょう。次のソースコードとブラウザでの表示を見てください。これはまだ、フロートを指定する前の段階です。

HTML

```
<div id="page">

<header>
header要素
</header>

<div id="main">
メインの段のテキストです。
メインの段のテキストです。
メインの段のテキストです。
メインの段のテキストです。
メインの段のテキストです。
メインの段のテキストです。
</div>

<div id="sub">
サブの段のテキストです。
サブの段のテキストです。
サブの段のテキストです。
</div>

<footer>
footer要素
</footer>

</div>
```

CSS

```
#page {
  margin: 0 auto;
  width: 300px;
}

header, footer {
  text-align: center;
  color: #fff;
  background: #bbb;       /* グレー */
}

#main {
  color: #fff;
  background: #fc0;       /* 黄色 */
}

#sub {
  color: #fff;
  background: #390;       /* 緑 */
}
```

2段組み前の状態のソースコード



上のソースコードを表示させたところ

コンテンツ全体は、「id="page"」が指定されたdiv要素でグループ化されています。そして、CSSでその幅を300ピクセルにし、左右のマージンを「auto」にすることでコンテンツ全体をセンタリングしています(「margin: 0 auto;」は上下のマージンを0、左右のマージンをautoにする指定です)。それ以外には、サンプルが見やすくなるようにヘッダーとフッターのテキストを中央揃えにして、各要素の背景色を指定しているだけです。これから2段組みにするのは、「id="main"」と「id="sub"」が指定されているdiv要素の部分です。これらのdiv要素は、各段の内容をグループ化している要素だと考えてください。

▶▶ floatプロパティを指定する

ではさっそくこれらの段にfloatプロパティを指定します。2つの段は幅の合計が300ピクセルになるように、幅は200ピクセルと100ピクセルにします。次の指定を追加してください。

CSS

```
・・・

#main {
  float: right;          追加
  width: 200px;
  color: #fff;
  background: #fc0;       /* 黄色 */
}

#sub {
  float: left;           追加
  width: 100px;
  color: #fff;
  background: #390;       /* 緑 */
}
```

source要素の使用例



この段階での表示。2段組みになった

ここではメインの段を右、サブの段を左に配置しましたが、もちろん左右を逆にしてもかまいませんし、このサンプルの場合は両方の段に「`float: right;`」を指定しても表示結果は同じになります(最後にフロートする段は余った場所に収まるしかないため、左右どちらを指定しても同じ場所に収まります)。ようするに、各段が自分が思った側に寄せられるように左右を指定してやればよいわけです。

重要なことは、すべての段にfloatプロパティを指定するということです。もし、フロートの状態になっていない段が1つでもあると、フロートしていない段のコンテンツは長さが短い段の下に入り込んでしまうことになります。

▶▶ 段にしないところの段組みを解除する

ところで、前ページのスクリーンショットを見ると、フッターがサブの段の下に入り込んでしまっています。これはフッターの前でフロートをクリアしていないことが原因です。次のようにフッターにclearプロパティを指定すると2段組みの完成です。

HTML　　sample/chapter-07/lecture-7-3/03.html

```
<div id="page">

<header>
header要素
</header>

<div id="main">
メインの段のテキストです。
メインの段のテキストです。
メインの段のテキストです。
メインの段のテキストです。
メインの段のテキストです。
メインの段のテキストです。
</div>

<div id="sub">
サブの段のテキストです。
サブの段のテキストです。
サブの段のテキストです。
</div>

<footer>
footer要素
</footer>

</div>
```

CSS　　sample/chapter-07/lecture-7-3/03-2cols.css

```
#page {
  margin: 0 auto;
  width: 300px;
}

header, footer {
  text-align: center;
  color: #fff;
  background: #bbb;        /* グレー */
}

#main {
  float: right;
  width: 200px;
```

```
  color: #fff;
  background: #fc0;       /* 黄色 */
}

#sub {
  float: left;
  width: 100px;
  color: #fff;
  background: #390;       /* 緑 */
}

footer {
  clear: both;
}
```

追加

footer要素にclearプロパティの指定を追加



フッターの直前でフロートがクリアされた

フロートによる3段組みレイアウト(1)

では、まったく同じ要領で3段組みを作ってみましょう。サブをもう1つ増やして、2段組みのときとまったく同じ要領で指定すると簡単に3段組みが実現できます。

HTML sample/chapter-07/lecture-7-5/04.html

```
<div id="page">

<header>
header要素
</header>
```

```
<div id="main">
メインの段のテキストです。
メインの段のテキストです。
メインの段のテキストです。
メインの段のテキストです。
メインの段のテキストです。
メインの段のテキストです。
メインの段のテキストです。
</div>

<div id="sub1">
サブ1の段のテキストです。
サブ1の段のテキストです。
サブ1の段のテキストです。
</div>

<div id="sub2">
サブ2の段のテキストです。
サブ2の段のテキストです。
サブ2の段のテキストです。
</div>

<footer>
footer要素
</footer>

</div>
```

CSS　sample/chapter-07/lecture-7-5/04-3cols-1.css

```
#page {
  margin: 0 auto;
  width: 400px;
}

header, footer {
  text-align: center;
  color: #fff;
  background: #bbb;
}

#main {
  float: right;
  width: 200px;
  color: #888;
  background: #eee;  /* 薄いグレー */
}

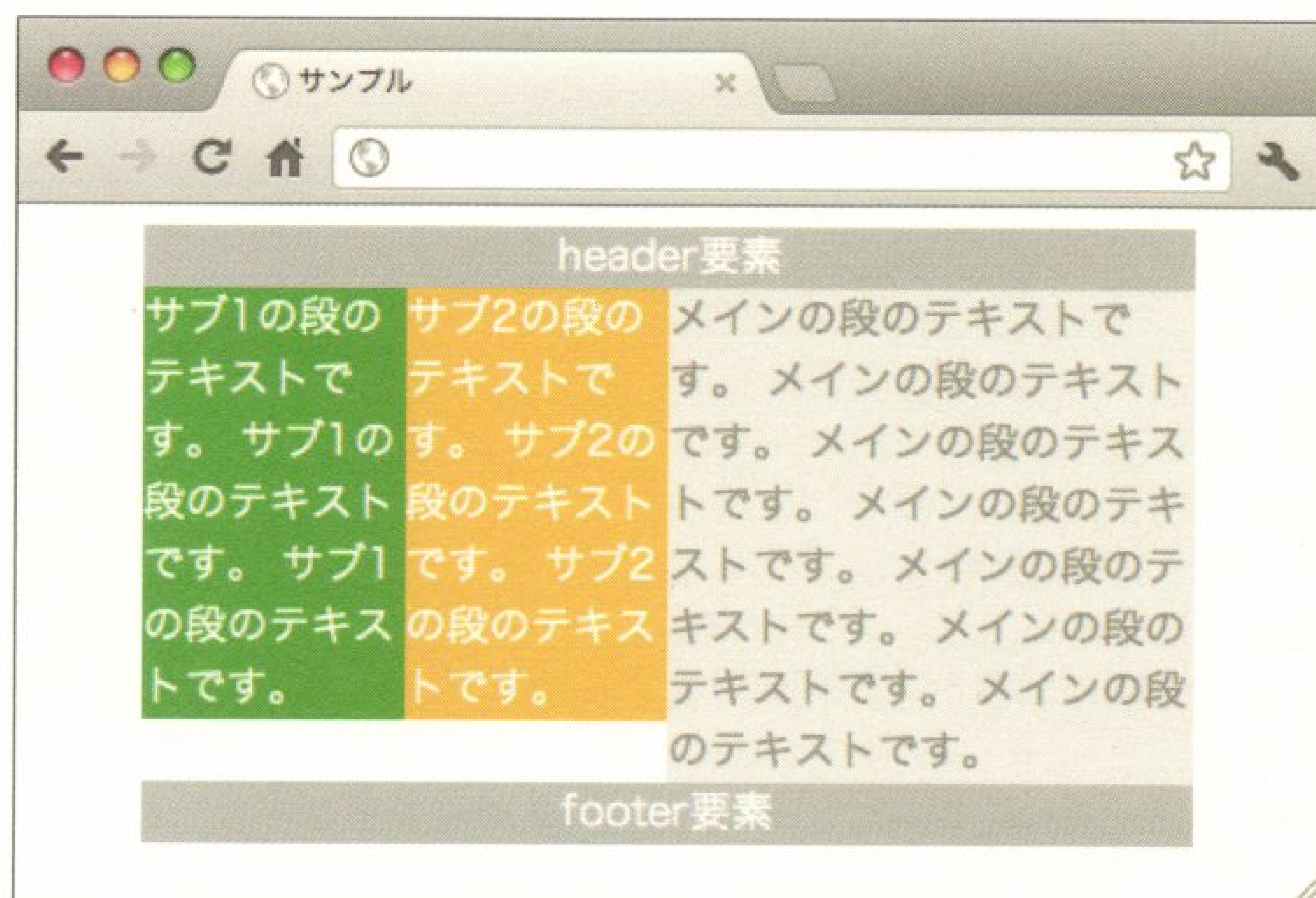
#sub1 {
  float: left;
  width: 100px;
  color: #fff;
  background: #390;  /* 緑 */
```

```
}

#sub2 {
  float: left;
  width: 100px;
  color: #fff;
  background: #fc0;  /* 黄色 */
}

footer {
  clear: both;
}
```

2段組みのときと同じ要領で3段組みを作成したソースコード



3段組みになった

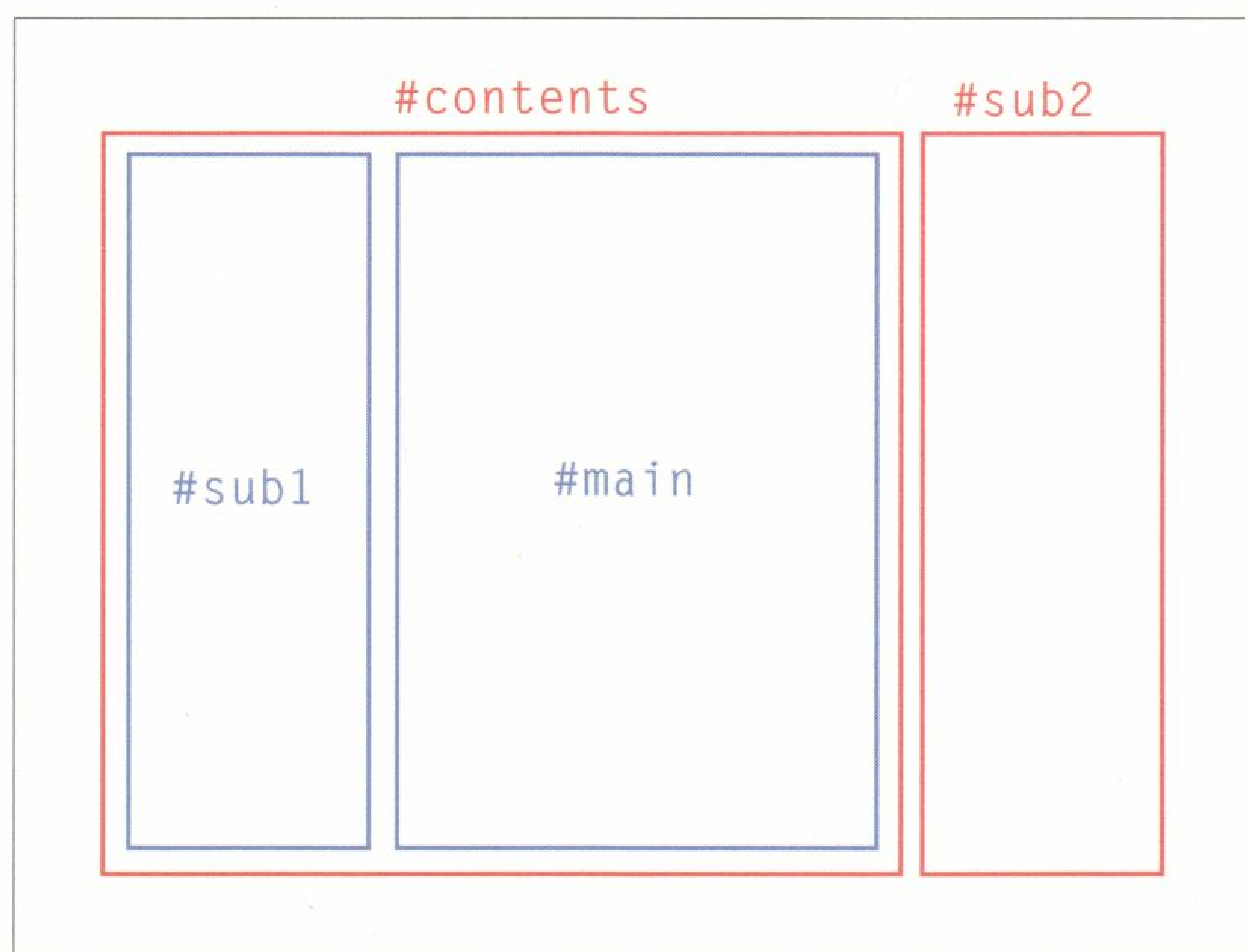
しかしここで、場合によっては問題が1つ発生します。このような指定方法では、HTML内で最初の段になっている**メインを3つの段の中央に配置できない**のです。最初にフロートを指定すると、その段はもっとも左またはもっとも右に配置されてしまうからです(同じ側にフロートを指定した場合、先に指定したものほど外側になり、あとから指定したものほど内側になります)。これはもちろんHTML上で順番を変更すれば可能ですが、HTML側では段の順序を変えたくない場合はどうすればいいのでしょうか。

次にそれを実現する3段組みの方法を紹介します。

フロートによる3段組みレイアウト(2)

▶▶ 2段組みの中に2段組みをつくる

メインを3つの段の中央に配置には、次のようにメインとそのとなりの段をdiv要素(下の例では#contents)でグループ化し、2段組みの一方の段の中がさらに2段組みになっているような状態にします。こうすることで、基本はすべて2段組みとなり、グループ化する段とフロートの方向次第でどの段をどの場所にでも配置できるようになります。



2つの段をグループ化することで、まず#contentsと#sub2の2段組み(赤で示した部分)があって、#contentsの中に#sub1と#mainの2段組み(青で示した部分)がある、というようにすべて2段組みにして扱えるようになる

では具体的なソースコードを見てみましょう。HTML側にはグループ化のための <div id="contents"></div> が追加されているだけです。CSS側ではそのdiv要素にfloatプロパティとwidthプロパティを指定しています。

HTML

```
<div id="page">

<header>
header要素
</header>

```

CHAPTER 7

```
<div id="contents">

<div id="main">
メインの段のテキストです。
メインの段のテキストです。
メインの段のテキストです。
メインの段のテキストです。
メインの段のテキストです。
メインの段のテキストです。
メインの段のテキストです。
</div>

<div id="sub1">
サブ1の段のテキストです。
サブ1の段のテキストです。
サブ1の段のテキストです。
</div>

</div>

<div id="sub2">
サブ2の段のテキストです。
サブ2の段のテキストです。
サブ2の段のテキストです。
</div>

<footer>
footer要素
</footer>

</div>
```

CSS

```
#page {
  margin: 0 auto;
  width: 400px;
}

header, footer {
  text-align: center;
  color: #fff;
  background: #bbb;
}

#main {
  float: right;
  width: 200px;
  color: #888;
  background: #eee;  /* 薄いグレー */
}

#sub1 {
  float: left;
```

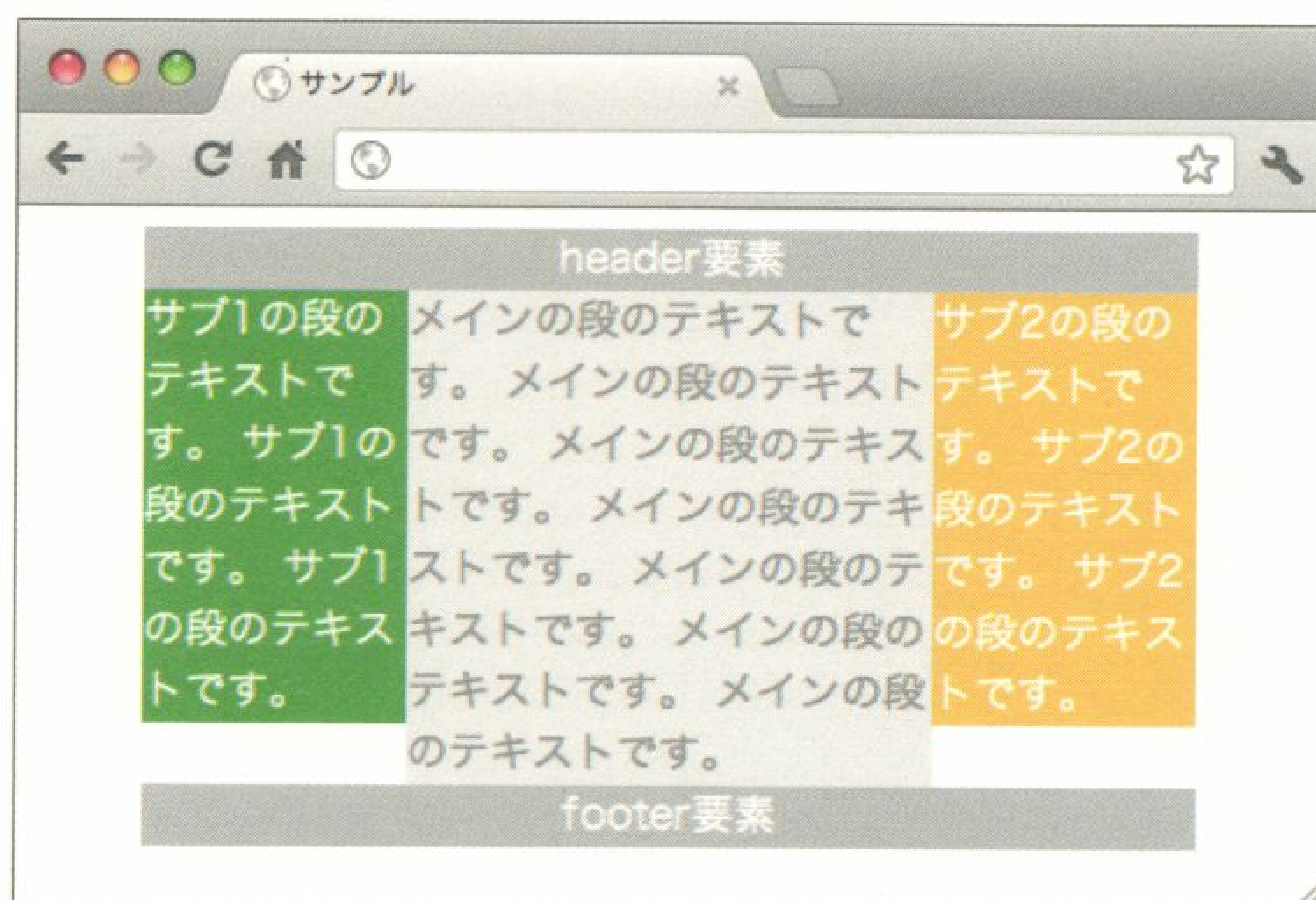
```
  width: 100px;
  color: #fff;
  background: #390;  /* 緑 */
}

#contents {
  float: left;
  width: 300px;
}

#sub2 {
  float: right;
  width: 100px;
  color: #fff;
  background: #fc0;  /* 黄色 */
}

footer {
  clear: both;
}
```

メインを3つの段の中央に配置するソースコード。3つの段のうち2つをグループ化している部分がポイント



上のソースコードを表示させると、メインの段が中央に表示される

▶▶ 段の高さを揃える

これでメインを中央に配置することはできましたが、スクリーンショットを見ると各段の高さが揃っていない点が気になります。しかし、フロートさせたコンテンツの高さを揃えることは、現時点で利用可能なCSSの範囲ではできません。そこで、このような場合には各段の背景を透明にし、コンテンツ全体をグループ化している要素にすべての段の分の背景を含んだ背景画像を表示させて、あたかも高さが揃っているように見せる方法がとられます。

これまでの例で言うと、#main、#sub1、#sub2の背景を消して、その代わりに#pageに次のような背景画像を表示させるということです。こうすることで、実際には高さはバラバラでも、高さが揃っているように見せることができます。



#pageに指定する背景画像。3つの段の色が入っている

HTML sample/chapter-07/lecture-7-5/05.html

```
<div id="page">

<header>
header要素
</header>

<div id="contents">

<div id="main">
メインの段のテキストです。
メインの段のテキストです。
メインの段のテキストです。
メインの段のテキストです。
メインの段のテキストです。
メインの段のテキストです。
メインの段のテキストです。
</div>

<div id="sub1">
サブ1の段のテキストです。
サブ1の段のテキストです。
サブ1の段のテキストです。
</div>

</div>

<div id="sub2">
サブ2の段のテキストです。
サブ2の段のテキストです。
サブ2の段のテキストです。
</div>

<footer>
footer要素
</footer>

</div>
```

CSS　sample/chapter-07/lecture-7-5/05-3cols-2.css

```
#page {
  margin: 0 auto;
  width: 400px;
  background: url(images/background.gif);   ← 追加
}

header, footer {
  text-align: center;
  color: #fff;
  background: #bbb;
}

#main {
  float: right;
  width: 200px;
  color: #888;   ← 背景の指定を削除
}

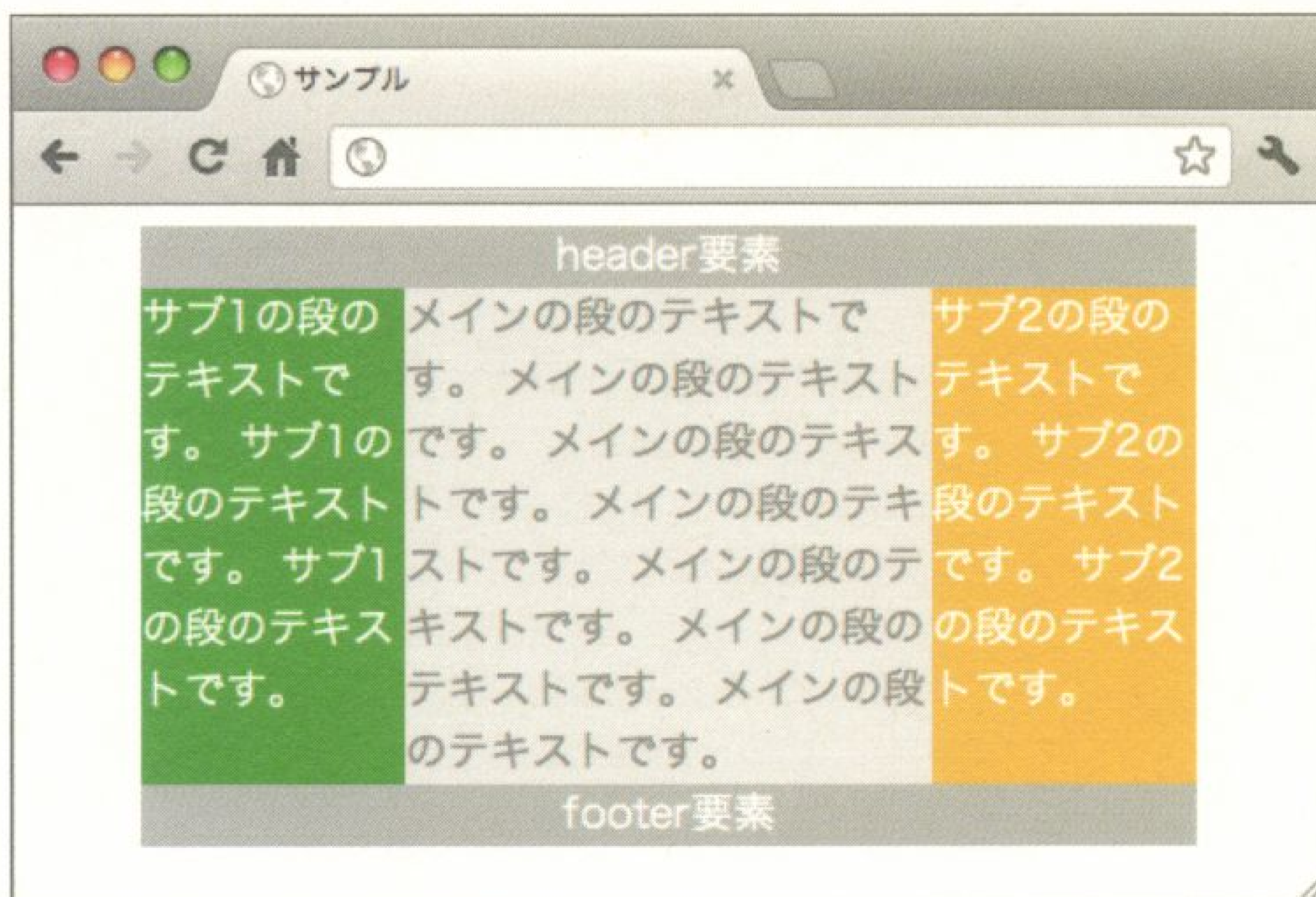
#sub1 {
  float: left;
  width: 100px;
  color: #fff;   ← 背景の指定を削除
}

#contents {
  float: left;
  width: 300px;
}

#sub2 {
  float: right;
  width: 100px;
  color: #fff;   ← 背景の指定を削除
}

footer {
  clear: both;
}
```

#pageに背景画像を表示させ、#main、#sub1、#sub2の背景を消す



前ページのソースコードを表示させたところ。高さが揃ったように見えている

相対配置と絶対配置

positionプロパティを使用すると、通常の配置方法とは異なる「相対配置」または「絶対配置」のモードに変更することができます。
「相対配置」は通常表示される位置から指定した距離だけ位置をずらす配置方法で、「絶対配置」は指定された要素を新しいレイヤーに移動させた上でbackground-positionプロパティのように表示位置を指定できる配置方法です。
いずれの配置モードでも、表示位置はtop、bottom、left、rightという4つのプロパティのうちのいずれかを使用して行います。まずはこれらに指定できる値を確認してから、それぞれのサンプルで具体的にどう表示されるのかを確認してみましょう。

positionに指定できる値

- static
  通常の配置モードにします。top、bottom、left、rightの各プロパティは、このモードでは無効となります。

- relative
  相対配置モードにします。このモードで配置位置を移動させても、他の要素の配置位置には一切影響を与えません。

- absolute
  絶対配置モードにします。この値が指定された要素は、元のレイヤーから取り除かれた状態となり、別の新しいレイヤーに配置されます。

positionに指定できる値（続き）

・fixed
固定配置モードにします。固定配置は絶対配置の一種ですが、位置指定の基準がbackground-attachmentプロパティと同様にページ全体となり、スクロールしても動かなくなります。Internet Explorer 6は、この値には対応していません。

top, bottom, left, rightに指定できる値

・単位つきの実数
表示位置を単位つきの実数で指定します。

・パーセンテージ
表示位置を基準となるボックスに対するパーセンテージで指定します。

・auto
状態に応じて自動的に調整します。

▶▶ 相対配置と絶対配置の表示例

では、これから相対配置と絶対配置をさせるサンプルの元の状態を見てみましょう。この段階ではまだCSSは指定していません。div要素の中にimg要素を3つ入れてあるだけです。ちなみに、ページ全体の上と左にある隙間は、body要素のマージン（ブラウザの初期設定）です。画像と画像の間に隙間があるのは、HTMLのソースコード上で入れてある改行が半角スペースに置き換わったもので、改行を入れなければ画像の隙間はなくなります。

HTML

```
<div>
<img id="pic1" alt="空の画像" src="images/sky.jpg">
<img id="pic2" alt="花の画像" src="images/yellow.jpg">
<img id="pic3" alt="草の画像" src="images/green.jpg">
</div>
```

相対配置と絶対配置の違いを確認するために用意されたソースコード



上のソースコードを表示させたところ。3つの画像が表示されている

相対配置の例

では、3つの画像のうちの真ん中の画像を相対配置にしてみましょう。positionプロパティの値に「relative」[※13]を指定し、topプロパティとleftプロパティで100pxを指定します(各画像の大きさは縦横200ピクセルです)。

HTML　sample/chapter-07/lecture-7-5/06.html

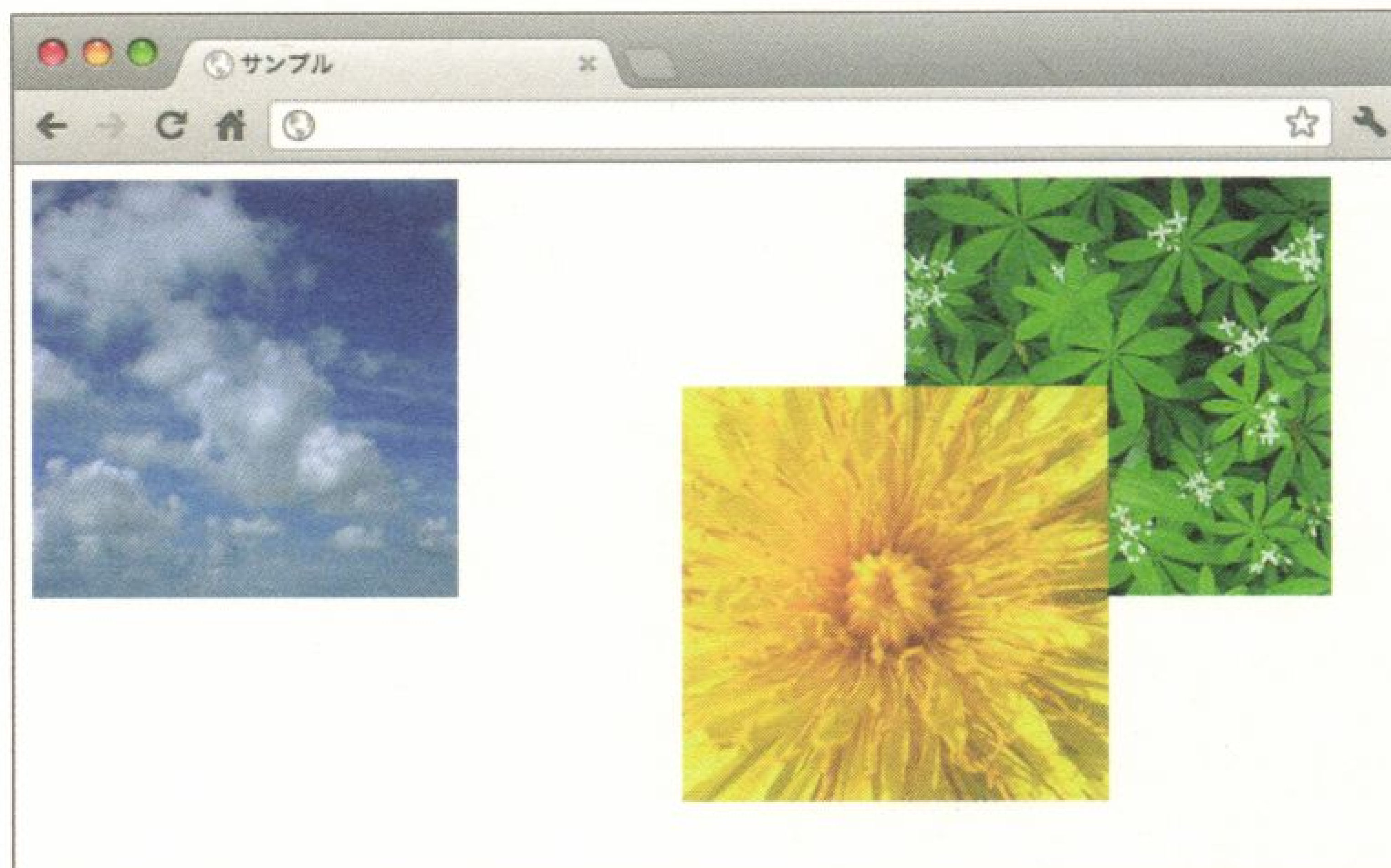
```
<div>
<img id="pic1" alt="空の画像" src="images/sky.jpg">
<img id="pic2" alt="花の画像" src="images/yellow.jpg">
<img id="pic3" alt="草の画像" src="images/green.jpg">
</div>
```

CSS　sample/chapter-07/lecture-7-5/06-position-1.css

```
#pic2 {
  position: relative;
  top: 100px;
  left: 100px;
}
```

相対配置にして、topを100ピクセル、leftを100ピクセルに指定

このCSSを適用してブラウザで表示させてみると次のようになります。まず、相対配置を指定していない要素には、まったく影響を与えていない点に注目してください。そして、真ん中の画像は100ピクセル下、100ピクセル右に移動しています。これが相対配置です。相対配置の場合、topは上から下方向への移動距離、leftは左から右方向への移動距離となるわけです。同様に、bottomは下から上方向への移動距離、rightは右から左方向への移動距離となります。



相対配置の表示結果

※13：「relative」の英語での発音は「レラティブ」です(「リレイティブ」ではありません)。

絶対配置の例

次は絶対配置です。topプロパティとleftプロパティはそのままにして、positionプロパティの値だけを「absolute」に変更して、表示がどう変わるのかを確認してみましょう。

HTML　sample/chapter-07/lecture-7-5/07.html

```
<div>
<img id="pic1" alt="空の画像" src="images/sky.jpg">
<img id="pic2" alt="花の画像" src="images/yellow.jpg">
<img id="pic3" alt="草の画像" src="images/green.jpg">
</div>
```

CSS　sample/chapter-07/lecture-7-5/07-position-2.css

```
#pic2 {
  position: absolute;   ← relativeからabsoluteに変更
  top: 100px;
  left: 100px;
}
```

さきほどのソースを絶対配置に変更

ブラウザで表示させてみると、絶対配置を指定していない要素にも**影響があった**ことが分かります。絶対配置にした要素は別レイヤーに移されたため、**元のレイヤーからは取り除かれた状態**となり、後続の要素の表示位置が変わっているのです。さらに、絶対配置を指定した要素の配置位置も、相対配置のときとは違っています。絶対配置を指定した要素は、**それを含む基準ボックスからの移動距離**で配置位置を指定するものだからです。この場合の基準ボックスはページ全体（html要素）で、topはその基準ボックスの上から絶対配置を指定したボックスの上までの距離、leftは基準ボックスの左から絶対配置を指定したボックスの左までの距離となります。同様に、bottomは下からの距離、rightは右からの距離となります。なお、さらに詳細に言うと基準ボックスのパディング領域の各上下左右から、絶対配置されたボックスのマージン領域の上下左右までの距離となります。



絶対配置の表示結果

絶対配置の基準ボックスは、自分を含むボックスのうち、positionプロパティの値として「relative」「absolute」「fixed」のいずれかが指定されている最も近い要素がなります。そのような要素がない場合は、html要素が基準ボックスとなります。特定の要素を基準ボックスにしたい場合は、その要素に「position: relative;」を指定するだけで基準ボックスにすることができます(その場合はtopやleftなどの指定は不要です)。

相対配置または絶対配置(固定配置も含む)になっている要素は、**z-indexプロパティ**を使ってそれらが重なる際の順序を指定することもできます。次の値が指定できます。

z-indexに指定できる値

- **整数**
  重なる順序を整数で指定します。大きい値ほど上に重なって表示されます。通常のコンテンツは0の状態となっています。負の値も指定できます。

- **auto**
  親要素と同じ階層にします。

COLUMN

絶対配置による段組み

Webページのレイアウトに本格的にCSSが使用されるようになってから数年間くらいまでは、絶対配置によって段組みレイアウトを行っているサイトも多くありました。しかし、絶対配置で段組みにすると、別レイヤーになったコンテンツと他のコンテンツはまったく影響しあわなくなるため、文字サイズを大きくした場合などにコンテンツの一部が重なって見えなくなってしまうなどの問題が発生することが分かってきました。つまり、大きさが変化したり、内容量が変化する可能性がある要素を絶対配置にすると、状況によっては下のレイヤーのコンテンツが見えなくなってしまう可能性があるのです。そのような理由から、現在では絶対配置にするのは、大きさやコンテンツの量が基本的には変化しない画像や、大きさが変化しても影響のない場所に配置するコンテンツなどにほぼ限定されているようです。

インライン要素の縦位置の指定

この章の最後に、配置と画像に関連するプロパティをもう1つだけ紹介しておきます。いきなりですが、次のサンプルを見てください。HTMLではh1要素の中にimg要素を入れ、CSSではh1要素に背景色（黄色）を指定しているだけのシンプルなソースコードです。

HTML　sample/chapter-07/lecture-7-5/08.html

```
<h1>
<img src="images/sky.jpg" alt="空の画像">
</h1>
```

CSS　sample/chapter-07/lecture-7-5/08-vertical-align.css

```
h1 { background: #fc0; }
```

黄色い背景の見出しの中に画像を入れただけのサンプル

これをブラウザで表示させると、次のようになります。ここで注目して欲しいのは、画像の下に隙間ができていることです。これはGoogle Chromeだからこうなっているというわけではなく、どのブラウザでも同じように隙間ができます。これが仕様通りの表示なのです。

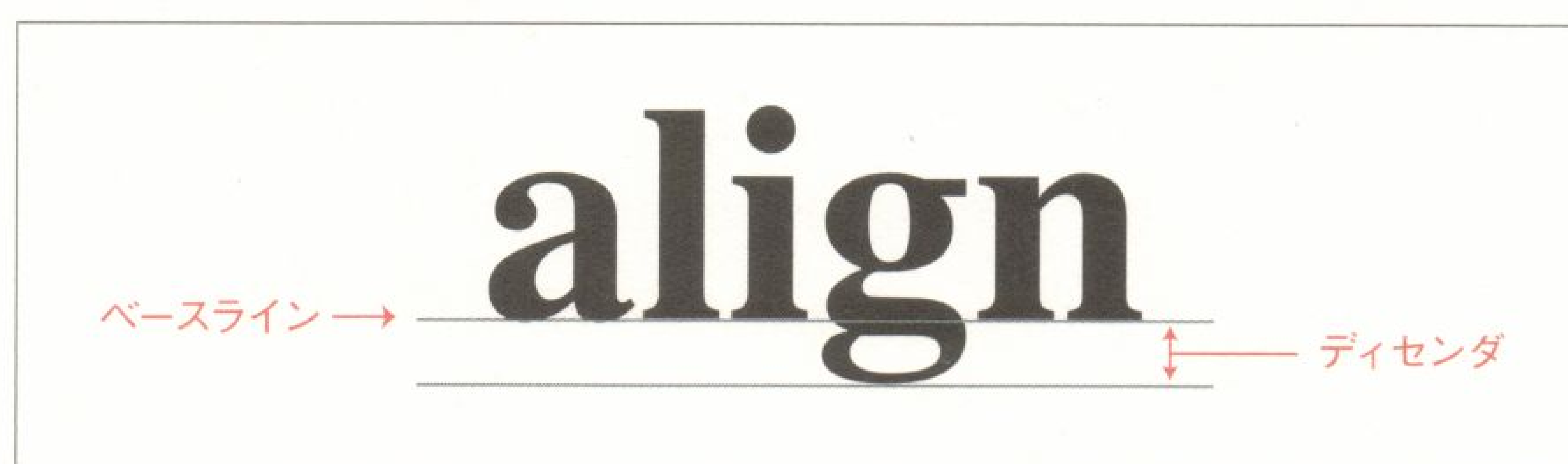
CHAPTER 7



上のサンプルの表示。画像の下に隙間ができている

▶▶ インライン要素の「ディセンダ」に注意

では、この隙間の正体は何で、それを消すためにはどうすればいいのでしょうか？　まずこの隙間の正体ですが、これは「ディセンダ」と呼ばれるもので、アルファベットの小文字の「g」や「j」のようにテキストのベースラインよりも下にはみ出す部分がある文字の、そのはみ出す部分を表示させるために用意されている領域です。img要素はインライン要素ですので、普通のテキストと同じように行の中に表示されているため、つまり、黄色い背景で表示されている領域は普通のインラインの1行であるために、このようにディセンダの領域も表示されているのです。もしh1要素の内容がテキストだけなら何の違和感もないのですが、テキストが一切なくて画像だけが入っているのでディセンダ領域が目立っているというわけです。



テキストのベースラインよりも下にはみ出す部分を表示させる領域をディセンダという

▶▶ vertical-alignプロパティ

CSSには、このようなインライン要素の縦方向の表示位置を調整するためのプロパティも用意されています。それが、**vertical-alignプロパティ**です。次の値が指定できます。

vertical-alignに指定できる値

- `baseline`
  ベースラインを親要素の行のベースラインに揃えます。画像のようにベースラインがない要素の場合は、その下をベースラインに揃えます。

- `top`
  上を揃えます。

- `middle`
  中央を揃えます。

- `bottom`
  下を揃えます。

- `super`
  上付き文字の表示位置に表示します。

vertical-alignに指定できる値（続き）

・sub
　下付き文字の表示位置に表示します。

・**単位つきの実数**
　親要素のベースラインからの距離を単位つきの実数で指定します。正の値は上方向、負の値は下方向への距離となります。

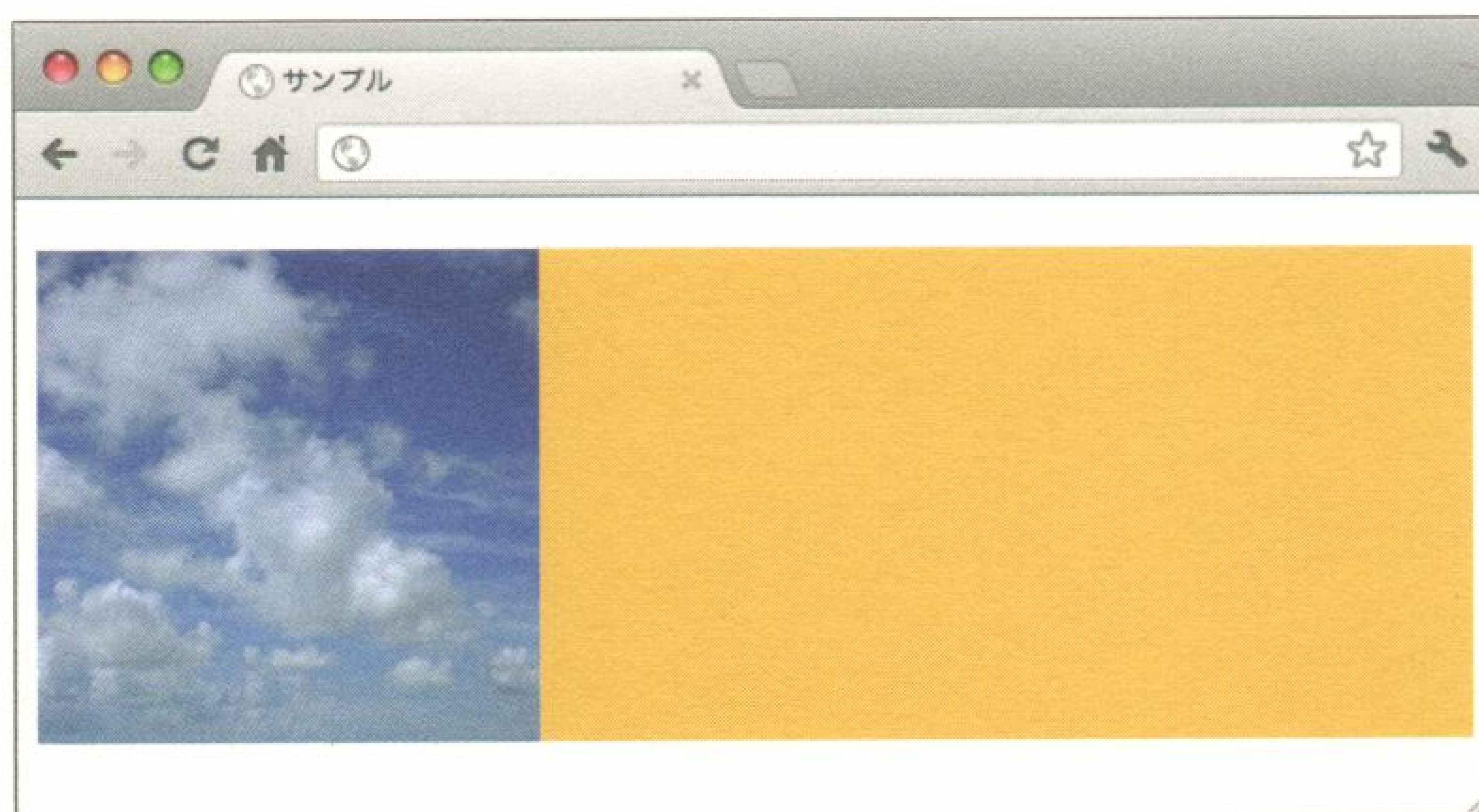
・**パーセンテージ**
　親要素のベースラインからの距離を、行の高さに対するパーセンテージで指定します。正の値は上方向、負の値は下方向への距離となります。

このプロパティの初期値は「baseline」であるため、特に何も指定しなければ画像は先程のサンプルのように表示されます。先程の画像に「vertical-align: bottom;」を指定すると、画像の下の隙間を消すことができます。

CSS

```
h1 { background: #fc0; }
img { vertical-align: bottom; }  ← この指定を追加
```

画像の下の隙間を消すには、このような指定を追加すればよい



上のサンプルの表示

CHAPTER 7

COLUMN

文字コードを指定しているのに文字化けする!?

文字コードは、HTMLの場合は「<meta charset="文字コード">」、CSSの場合は「@charset "文字コード";」のように指定します。しかし、この通りの書式でしっかりと文字コードを指定しているにもかかわらず、文字化けが発生してしまう場合があります。

たとえば、実際に保存されている文字コードと、HTMLやCSSで指定している文字コードが違っている場合には文字化けが発生します。また、環境によっては、ファイルをサーバーに転送する際に、文字コードが自動的に別のものに変換されてしまうというケースもあり、その際にも文字化けが発生します。文字コードをきちんと指定しているにもかかわらず文字化けする場合は、HTMLやCSSで指定している文字コードと実際に保存されている文字コードが同じになっているかを確認してみてください。

CHAPTER 8

ナビゲーション

Chapter 7までの内容を覚えていれば、一般的なページの多くの部分を作ることができます。しかし、ナビゲーションだけはちょっと特殊でその例外となります。Chapter 8では、ナビゲーションに関連する要素やプロパティをまとめて紹介します。

LECTURE 8-1

ナビゲーションに関連する要素

グローバルナビゲーションのようなページ内の主要なナビゲーションに対して使用する要素について説明します。

ナビゲーションのセクション 【HTML5新】

Chapter 7では、4つあるセクション関連要素のうち、3つだけを解説しました。残りの1つが、ここで説明するnav要素です。

要素名	意味
section	セクション
article	内容がそれだけで完結しているセクション
aside	主コンテンツには含まれないセクション
nav	主要なナビゲーションのセクション

HTML5のセクションをあらわす要素、全4種類

nav要素はその部分が**ナビゲーションのセクション**であることを示す要素ですが、すべてのナビゲーションをこの要素としてマークアップすべきかというと、そうではありません。
この要素は、たとえば一般的なグローバル・ナビゲーションのように、ページ内での**主要なナビゲーション部分に対してのみ**使用します。よくあるフッター部分のリンクなどに対して使用するものではありません。あくまで、ナビゲーションの**セクション**に対して使用するものである点に注意してください。

リスト関連の要素

▶▶ 3種類のリスト要素

ここでいうリスト関連の要素とは、箇条書きのような形式でテキストを表示させる要素のことで、直接的にはナビゲーションとは関係ありません。しかし、ナビゲーションの各項目を（表示形式とは関係なく意味的な面から）どの要素としてマークアップするかということを考えたときに、この要素以上にぴったりと当てはまる要素はほかにありません。そのため、ナビゲーション部分をマークアップするには、一般的にここで紹介するul要素が使用されています。ここでは、そのul要素をはじめとする3種類の要素について説明します。

要素名	意味
ul	箇条書きタイプのリスト（範囲全体）
ol	番号付きタイプのリスト（範囲全体）
li	リスト内の1項目

箇条書きタイプのリスト関連要素

まず、箇条書きタイプのリストには2種類があります。リスト内の各項目に番号がついているタイプとついていないタイプです。通常、ナビゲーションに使用されるのはその「番号のついていないタイプ」の方で、英語の「unordered list」の先頭の文字をそれぞれとって**ul要素**という名前になっています。「番号のついているタイプ」は英語では「ordered list」となるため、**ol要素**と呼ばれています。

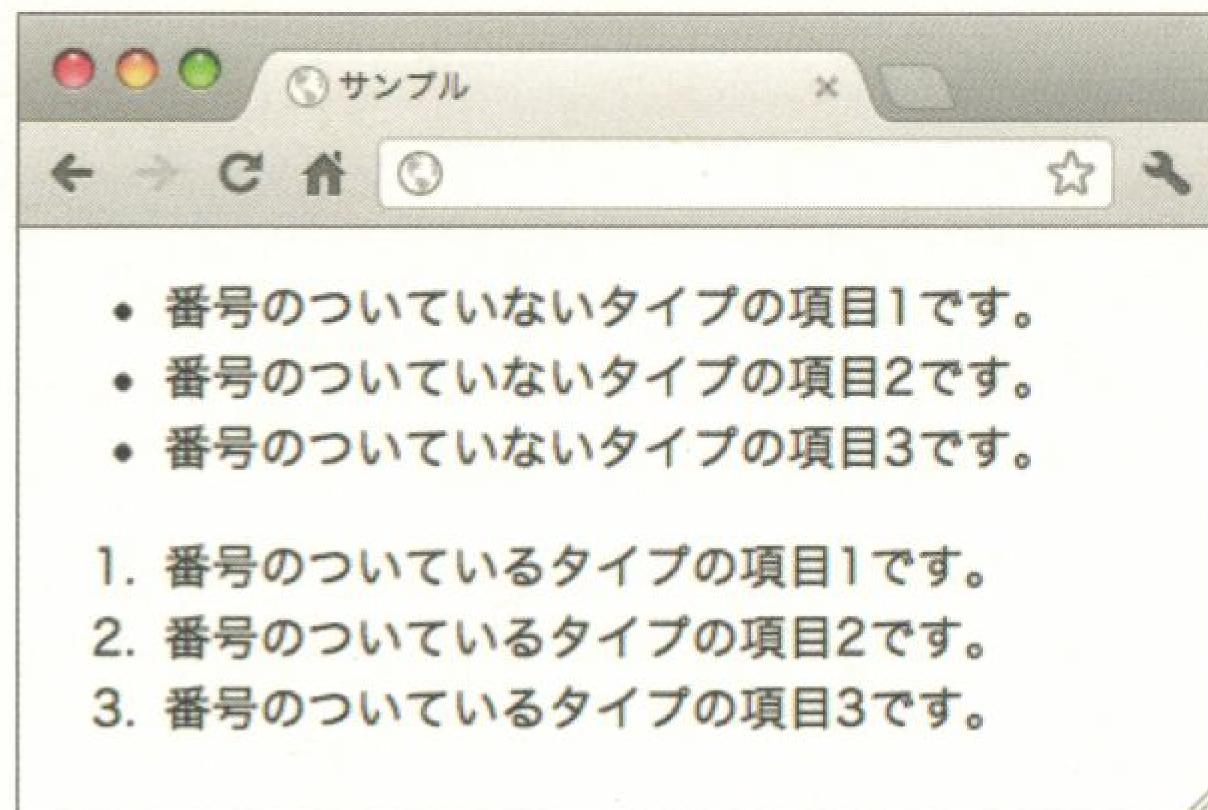
ul要素とol要素は、それぞれのリストの範囲全体を示すもので、その中の各項目は**li要素**としてマークアップします。li要素の「li」は「list item」の略です。ここで簡単な例を示しますので、タグのつけ方と基本的にどのように表示されるのかを確認してください。

HTML　　**sample/chapter-08/lecture-8-1/01.html**

```
<ul>
<li>番号のついていないタイプの項目1です。</li>
<li>番号のついていないタイプの項目2です。</li>
<li>番号のついていないタイプの項目3です。</li>
</ul>

<ol>
<li>番号のついているタイプの項目1です。</li>
<li>番号のついているタイプの項目2です。</li>
<li>番号のついているタイプの項目3です。</li>
</ol>
```

ul要素とol要素のマークアップの例



前ページのソースコードの表示例

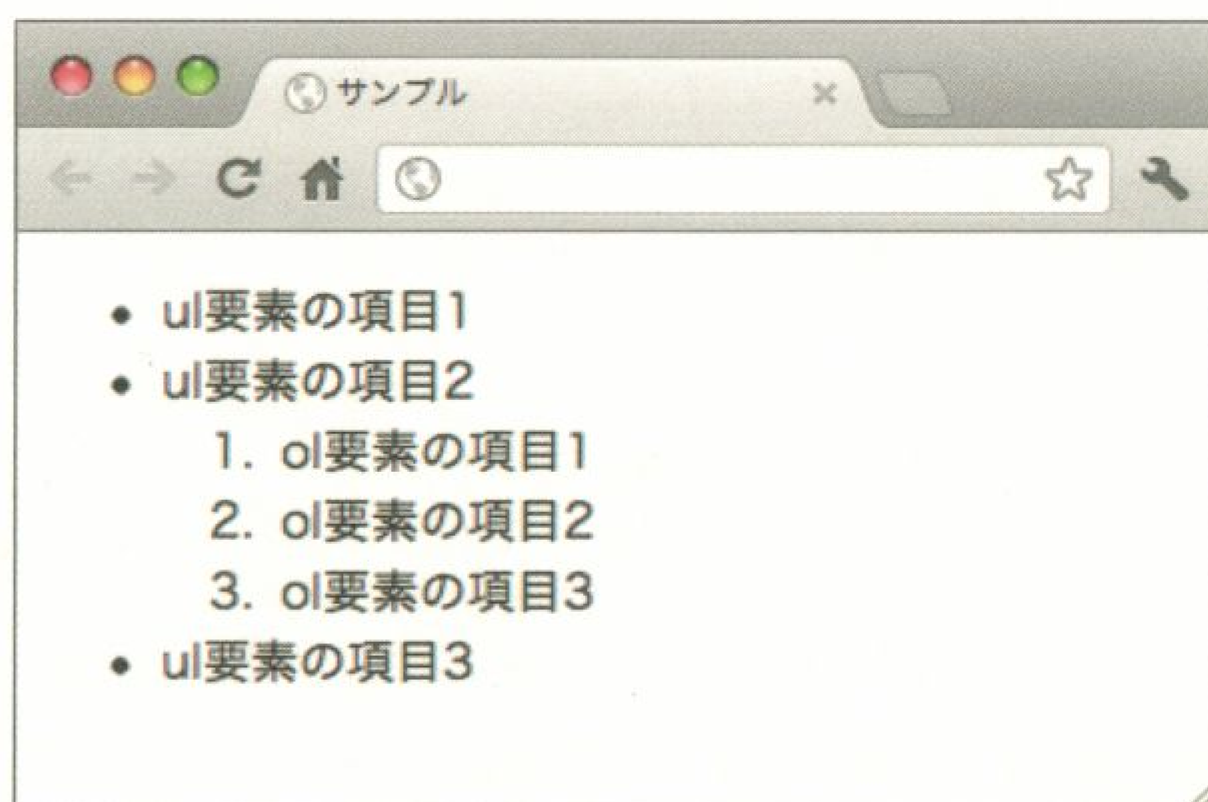
▶▶ ul要素とol要素の子要素

ul要素とol要素に直接の子要素として入れられるのは、li要素だけです。したがって、それらの要素を入れ子にする場合には、ul要素またはol要素内に直接別のul要素またはol要素内を入れるのではなく、li要素の中に入れるようにしてください。li要素の中にはインライン要素でもブロックレベル要素でも自由に入れることができます。

HTML　　**sample/chapter-08/lecture-8-1/02.html**

```
<ul>
  <li>ul要素の項目1</li>
  <li>ul要素の項目2
    <ol>
      <li>ol要素の項目1</li>
      <li>ol要素の項目2</li>
      <li>ol要素の項目3</li>
    </ol>
  </li>
  <li>ul要素の項目3</li>
</ul>
```

リストを入れ子にする場合のマークアップの例。入れ子にするul要素やol要素は、li要素の中に入れる必要がある



入れ子にした場合の表示例

なお、ol要素およびol要素内のli要素には、次の属性が指定できます。

ol要素に指定できる属性

- **start="開始番号"**
  番号を1以外から始めたい場合に、その先頭の番号を整数で指定します。

- **type="番号の種類"**
  各項目に表示させる番号の種類を指定します。次の5種類の値が指定できます。

値	番号の種類	表示例
1	整数	1. 2. 3.
a	アルファベット(小文字)	a. b. c.
A	アルファベット(大文字)	A. B. C.
i	ローマ数字(小文字)	i. ii. iii.
I	ローマ数字(大文字)	I. II. III.

li要素に指定できる属性

- **value="番号"**　※ol要素の子要素である場合のみ指定可
  ol要素内のその項目の番号を整数で指定します。この項目のあとに続く項目の番号は、ここで指定した番号に続く連番となります。

用語説明型のリスト 【HTML5改】

ナビゲーション部分に使われることはあまりありませんが、リストに分類される要素として**用語説明型のリスト**も紹介しておきましょう。用語説明型のリストとは、箇条書きのように単純に項目が複数あるタイプのリストではなく、**用語とその説明のように内容の項目がペアになっている**タイプのリストです。用語説明型のリストとは言っても、用語とその説明に限らず、質問と回答のように内容が対になっている形式のデータ全般に対して使用することが可能です。

要素名	意味
dl	用語説明型のリスト(範囲全体)
dt	リスト内の「用語」に該当する部分
dd	リスト内の「説明」に該当する部分

用語説明型リストの関連要素

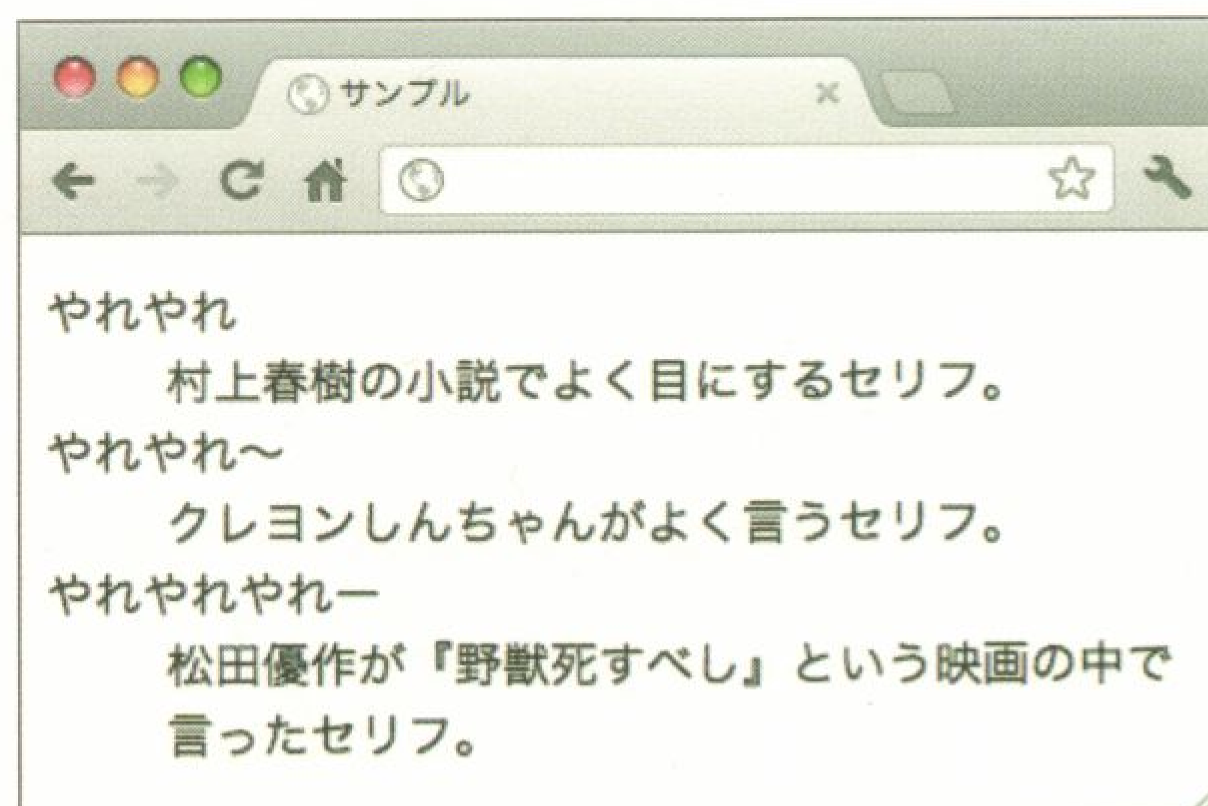
ul要素とol要素のように、用語説明型のリストの全体を示すのが**dl要素**です。dlは「description list」の略です。そして、その中にli要素のように入れる要素が、用語説明型のリストの場合は2種

類あります。「用語(term)」部分に該当する**dt要素**と、「説明文(description)」部分に該当する**dd要素**です。では、具体的な使い方の例を紹介しましょう。

HTML　sample/chapter-08/lecture-8-1/03.html

```
<dl>
  <dt>やれやれ</dt>
  <dd>村上春樹の小説でよく目にするセリフ。</dd>
  <dt>やれやれ〜</dt>
  <dd>クレヨンしんちゃんがよく言うセリフ。</dd>
  <dt>やれやれやれー</dt>
  <dd>松田優作が『野獣死すべし』という映画の中で言ったセリフ。</dd>
</dl>
```

dl要素のマークアップ例



上のソースコードの表示例

dl要素内には、上の例のようにdt要素とdd要素をペアにして複数入れることができます。ペアといってもdt要素とdd要素は常に1つずつである必要はなく、dt要素またはdd要素を連続して複数入れることも可能です。ただし、同じペア内ではdt要素は必ずdd要素よりも前に配置する必要があります。

COLUMN

dl要素はもともとは「定義リスト」だった!?

dl要素の「dl」が「description list」の意味になったのはHTML5からです。それ以前は、「definition list」の意味で「定義リスト」と呼ばれていました。簡単に言えば、dl要素はもともとは専門用語などを定義する部分で使用するために用意された要素だったのです。とはいえ、使用する対象はそれだけに限定されていたわけでもありませんので、使い方が変わったということではありません。HTML5になって、要素名の意味がより汎用的なものに修正されたということです。

LECTURE 8-2

リスト関連のプロパティ

ナビゲーションを作成する際に必須というわけではないのですが、CSSにはul要素とol要素に使用できる専用のプロパティが用意されていますのでここで紹介しておきます。

プロパティ名	機能
list-style-type	行頭記号を変える
list-style-image	行頭記号を画像にする
list-style-position	行頭記号の表示位置を設定する
list-style	リスト関連プロパティの一括指定

リスト関連のプロパティ

なお、ここで紹介するプロパティはすべて行頭記号に関するものですので、指定する対象はli要素となります。しかし、これらのプロパティはすべてその内部の要素にも値を継承するタイプとなっているため、ul要素やol要素に対して指定すると、その子要素であるli要素にも適用されることになります。

行頭記号を変える

list-style-typeプロパティは、行頭記号を変更するプロパティです。次の値が指定できます。表示される行頭記号は、ブラウザの種類によって異なる場合があります。

list-style-typeに指定できる値

・none
　行頭記号を消します。

・disc
　塗りつぶした丸にします。

・circle
　白抜きの丸にします。

・square
　四角にします。

list-style-typeに指定できる値（続き）

- **decimal**
  数字にします。

- **decimal-leading-zero**
  01. 02. 03. ～ 99. のように先頭に0をつけた数字にします。

- **lower-roman**
  小文字のローマ数字にします。

- **upper-roman**
  大文字のローマ数字にします。

- **lower-latin**
  小文字のアルファベットにします。

- **upper-latin**
  大文字のアルファベットにします。

- **lower-alpha**
  小文字のアルファベットにします。

- **upper-alpha**
  大文字のアルファベットにします。

- **lower-greek**
  小文字のギリシャ文字にします。

HTML　sample/chapter-08/lecture-8-2/01.html

```html
<ul id="sample1">
<li>項目1</li>
<li>項目2</li>
<li>項目3</li>
</ul>

<ul id="sample2">
<li>項目1</li>
<li>項目2</li>
<li>項目3</li>
</ul>

<ul id="sample3">
<li>項目1</li>
<li>項目2</li>
<li>項目3</li>
</ul>

```

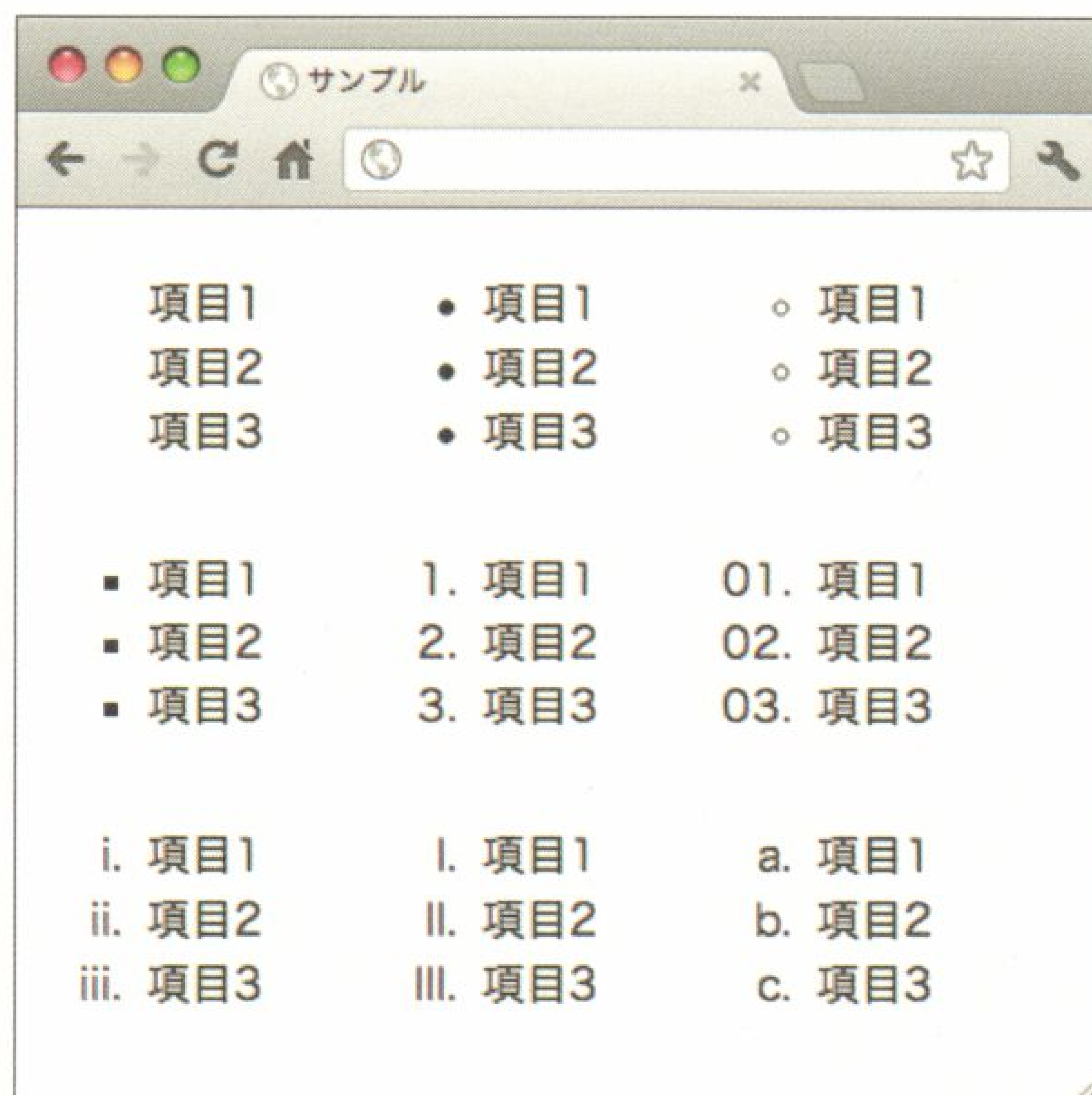
```
…中略…

<ol id="sample9">
<li>項目1</li>
<li>項目2</li>
<li>項目3</li>
</ol>
```

CSS　sample/chapter-08/lecture-8-2/01-list-style-type.css

```
ul, ol {
  float: left;
  margin-right: 40px;
}
#sample1 { list-style-type: none; }
#sample2 { list-style-type: disc; }
#sample3 { list-style-type: circle; }
#sample4 { list-style-type: square; }
#sample5 { list-style-type: decimal; }
#sample6 { list-style-type: decimal-leading-zero; }
#sample7 { list-style-type: lower-roman; }
#sample8 { list-style-type: upper-roman; }
#sample9 { list-style-type: lower-alpha; }
```

list-style-typeプロパティの指定例



上のソースコードの表示例

行頭記号を画像にする

list-style-imageプロパティを使用すると、行頭記号を画像に変えることができます。指定できる値は次の通りで、画像はurl(～)の書式で指定します。

list-style-imageに指定できる値

・**url(画像のアドレス)**
　指定した画像を行頭記号として表示させます。

・**none**
　画像を行頭記号として表示させません。

HTML　sample/chapter-08/lecture-8-2/02.html

```html
<ul>
<li>項目1</li>
<li>項目2</li>
<li>項目3</li>
</ul>
```

CSS　sample/chapter-08/lecture-8-2/02-list-style-image.css

```css
ul { list-style-image: url(triangle.gif); }
```

list-style-imageプロパティの指定例



上のソースコードの表示例

行頭記号の表示位置を設定する

list-style-position プロパティを使用すると、行頭記号の表示位置を、テキストを表示させる領域の先頭部分に変更することができます。次の値が指定できます

list-style-positionに指定できる値

- `outside`
  テキストを表示させる領域の外側に行頭記号を表示させます。

- `inside`
  テキストを表示させる領域の内側に行頭記号を表示させます。

行頭記号の表示位置を変更しても、項目内のテキストが1行だけだと違いはよく分かりません。しかし、下のサンプルのようにテキストが複数行になっていると、行頭記号の位置の違いがはっきりと分かります。

HTML　sample/chapter-08/lecture-8-2/03.html

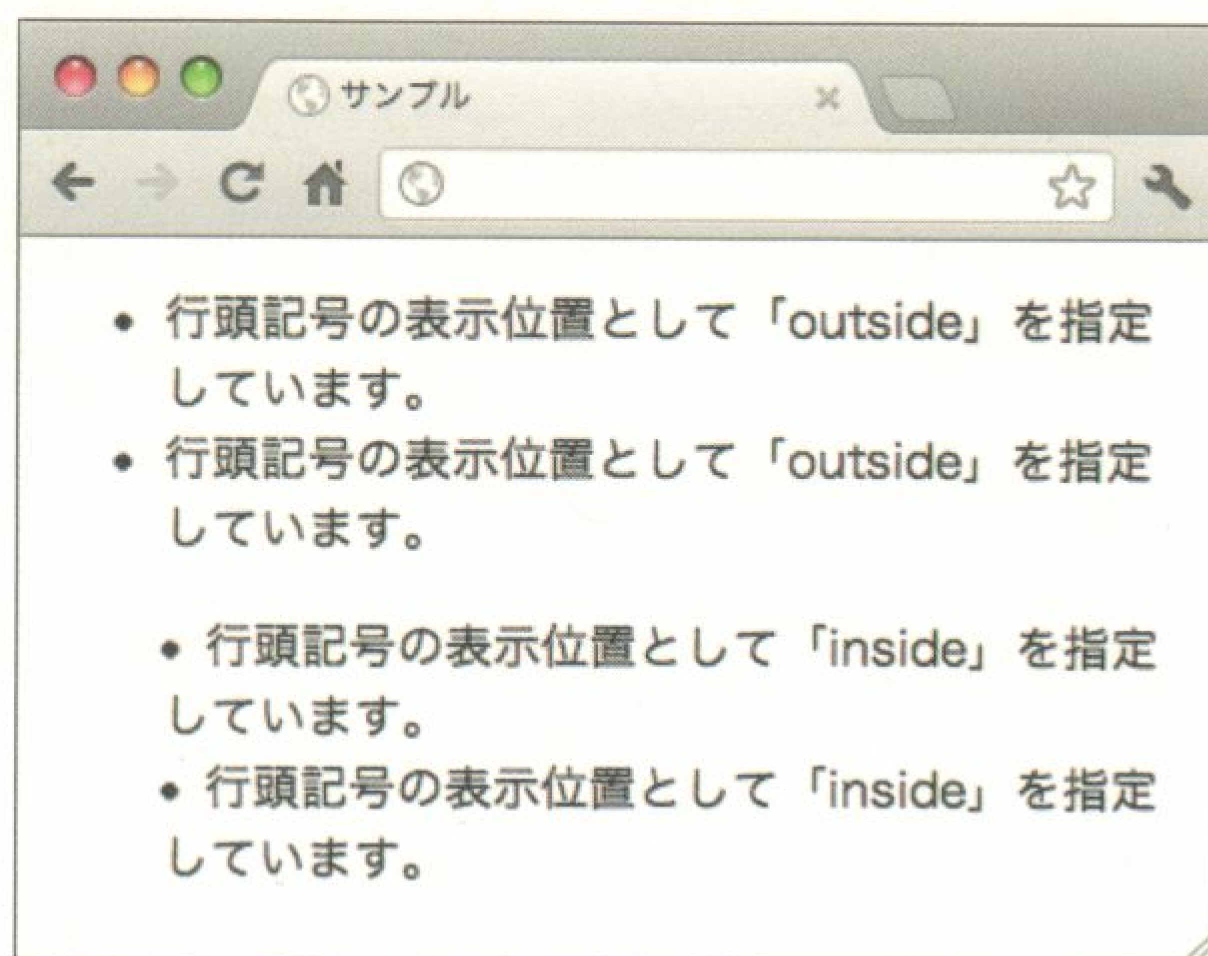
```
<ul id="sample1">
<li>行頭記号の表示位置として「outside」を指定しています。</li>
<li>行頭記号の表示位置として「outside」を指定しています。</li>
</ul>

<ul id="sample2">
<li>行頭記号の表示位置として「inside」を指定しています。</li>
<li>行頭記号の表示位置として「inside」を指定しています。</li>
</ul>
```

CSS　sample/chapter-08/lecture-8-2/03-list-style-position.css

```
#sample1 { list-style-position: outside; }
#sample2 { list-style-position: inside; }
```

list-style-position プロパティの指定例



前ページのソースコードの表示例

リスト関連プロパティの一括指定

list-styleプロパティを使用すると、リスト関連プロパティの値を任意の順でまとめて一括で指定できます。必要な値を半角スペースで区切って指定します。「none」を指定すると、list-style-typeとlist-style-imageの両方の値が「none」に設定されます。

list-styleに指定できる値

- **list-style-typeの値**
  list-style-typeに指定できる値が指定できます。
- **list-style-imageの値**
  list-style-imageに指定できる値が指定できます。
- **list-style-positionの値**
  list-style-positionに指定できる値が指定できます。

HTML sample/chapter-08/lecture-8-2/04.html

```
<ul>
<li>項目1</li>
<li>項目2</li>
<li>項目3</li>
</ul>
```

CSS sample/chapter-08/lecture-8-2/04-list-style.css

```
ul { list-style: url(triangle.gif) square; }
```

list-styleプロパティの指定例



前ページのソースコードの表示例

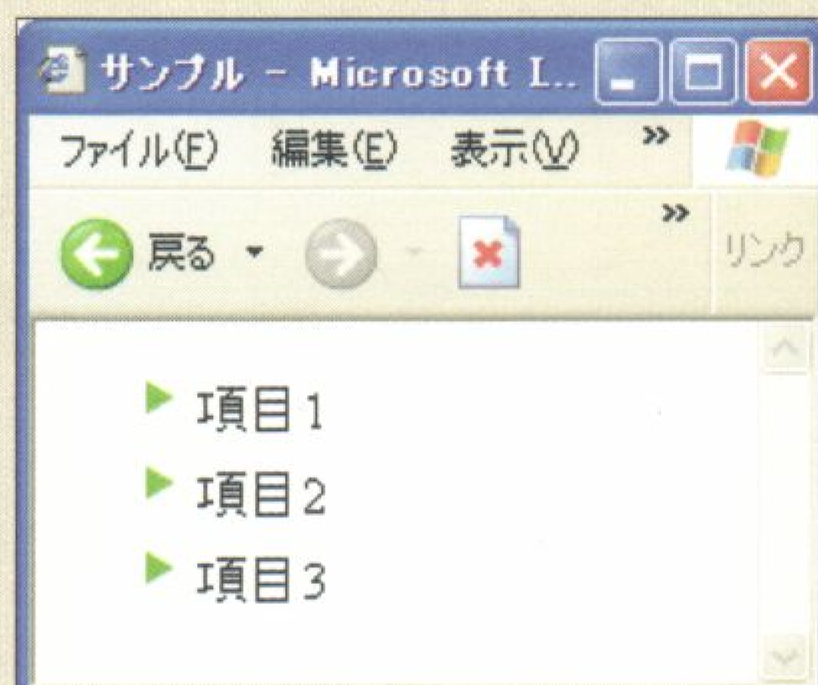
COLUMN

行頭記号を画像にすると位置がずれる!?

Internet Explorer 7以前では、次のように行間を広くすると画像とテキストの表示位置がずれてしまいます。

```
ul {
  line-height: 1.7;
  list-style-image: url(triangle.gif);
}
```

line-heightプロパティで行間を1.7（フォントサイズの1.7倍）に設定

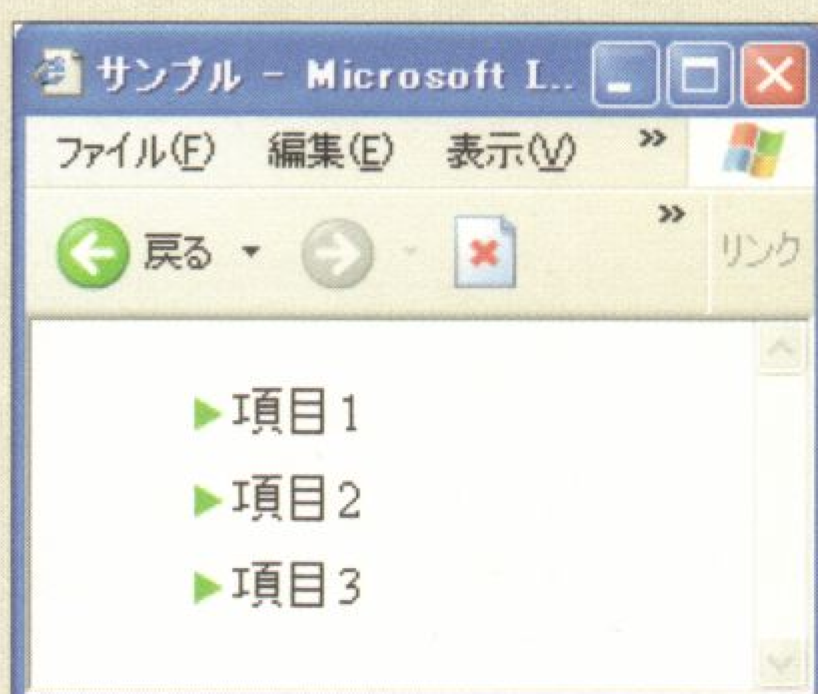


Internet Explorer 6での表示結果。画像とテキストの表示位置がずれている

ところが残念なことに、CSSには画像にした行頭記号の位置を調整するプロパティは用意されていません。そのため、このような場合は一般に、行頭記号はあえて消して、表示位置の調整が可能な背景画像を代わりに表示させるという、次ページのような対策がとられています。

```
ul {
  line-height: 1.7;
  list-style: none;
}
li {
  padding-left: 12px;
  background: url(triangle.gif) no-repeat left center;
}
```

list-styleプロパティで行頭記号を消し、行頭記号は背景画像として表示させると表示位置が調整可能となる



Internet Explorer 6での表示結果。位置のずれは目立たなくなっている

ただし上の例のように指定していると、たとえば項目内のテキストが2行になった場合は、行頭記号は1行目と2行目の間に表示されることになります。複数行にも対応させる場合は、背景画像の表示位置が1行目に合うように変更する必要があります。

LECTURE 8-3

表示形式を変えるプロパティ

今度は逆に、リストと特別に関連が深いわけではないのですが、ナビゲーションを作成する際によく使われる「表示形式を変えるプロパティ」について説明します。

表示形式を変更する

displayプロパティは、ナビゲーションを作成する際に欠かせないプロパティの1つです。このプロパティを使用すると、インライン要素をブロックレベル要素のように表示させたり、ブロックレベル要素をインライン要素のように表示させることなどができます。指定できる表示形式（値）は、次の通りです。

displayに指定できる値

・inline
インライン要素と同様の表示にします。

・block
ブロックレベル要素と同様の表示にします。

・inline-block
ボックス自体はインライン要素と同様に配置されますが、その内部はブロックレベル要素のように複数行を表示できるボックスにします（フォームのテキスト入力欄で複数行を入力できるタイプのものと同様の表示形式になります）。

・none
ボックスを消します（ボックスが無い状態になります）。

▶▶ displayプロパティの使用例

では、このプロパティを使用すると、具体的にどのように表示が変化するのかを見てみましょう。次のサンプルのHTMLファイルでは、インライン要素であるspan要素とブロックレベル要素であ

CHAPTER 8

るdiv要素のペアを4つ配置しています。それぞれには、sample1〜sample4というクラス名をつけています。

HTML　　sample/chapter-08/lecture-8-3/01.html

```
<span class="sample1">サンプル1<br>span要素</span>
<div class="sample1">サンプル1<br>div要素</div>

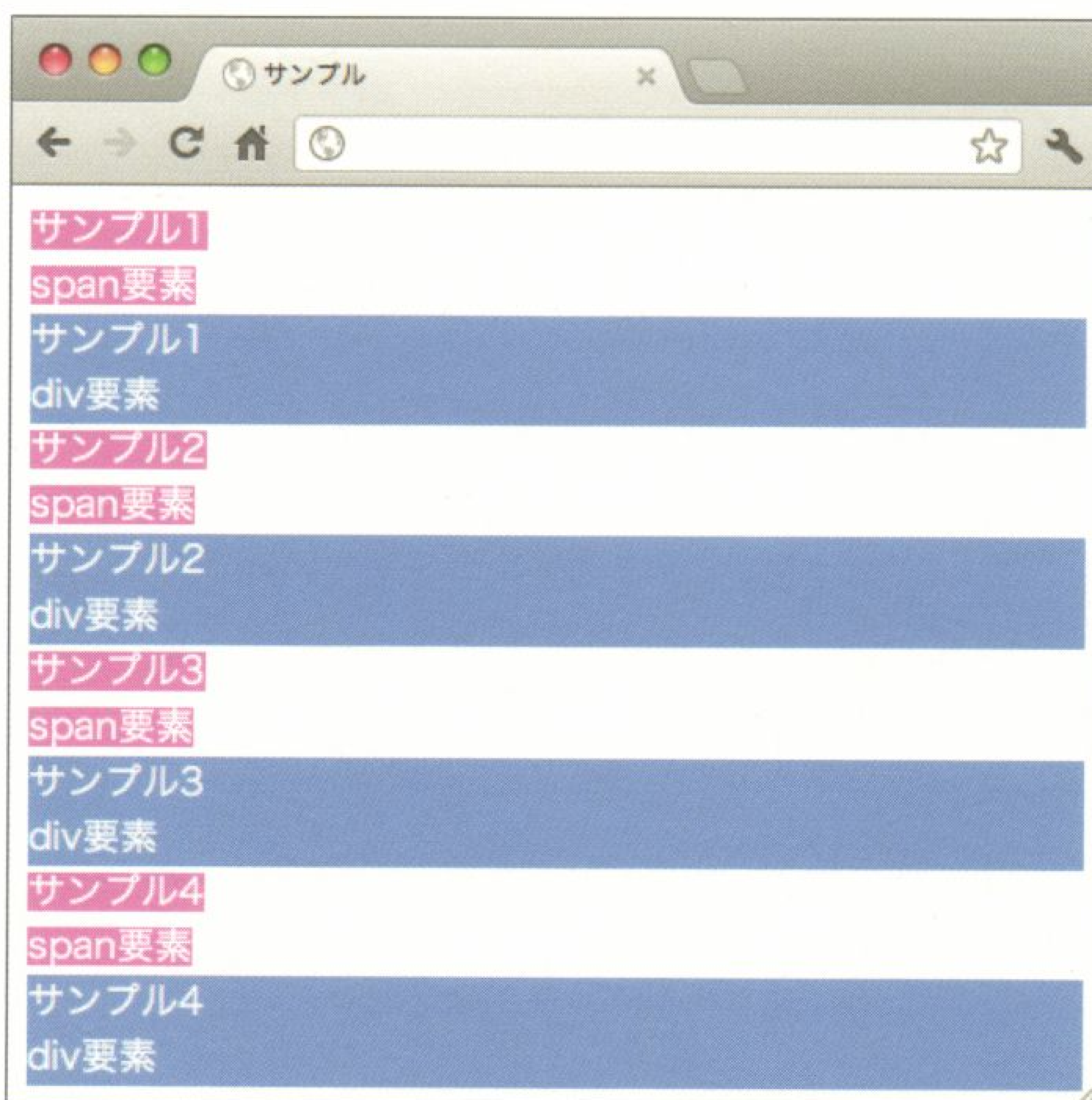
<span class="sample2">サンプル2<br>span要素</span>
<div class="sample2">サンプル2<br>div要素</div>

<span class="sample3">サンプル3<br>span要素</span>
<div class="sample3">サンプル3<br>div要素</div>

<span class="sample4">サンプル4<br>span要素</span>
<div class="sample4">サンプル4<br>div要素</div>
```

サンプルのHTMLソース。span要素とdiv要素のペアが4つある

まずはdisplayプロパティを指定していない状態の表示を確認しておきましょう。span要素とdiv要素の要素内容にはbr要素を入れて意図的に改行させてありますので、このような表示になっています。



displayプロパティを指定していない状態での表示

これに対してdisplayプロパティを次のように指定すると、表示結果はこのように変わります。まず、サンプル1のspan要素とdiv要素はともにインライン要素の表示になっています。同様に、サンプル2は両方ともブロックレベル要素の表示になっています。サンプル3は消えてなくなり、サンプル2の下にはサンプル4のボックスが表示されています。サンプル4のように、インライン要素と

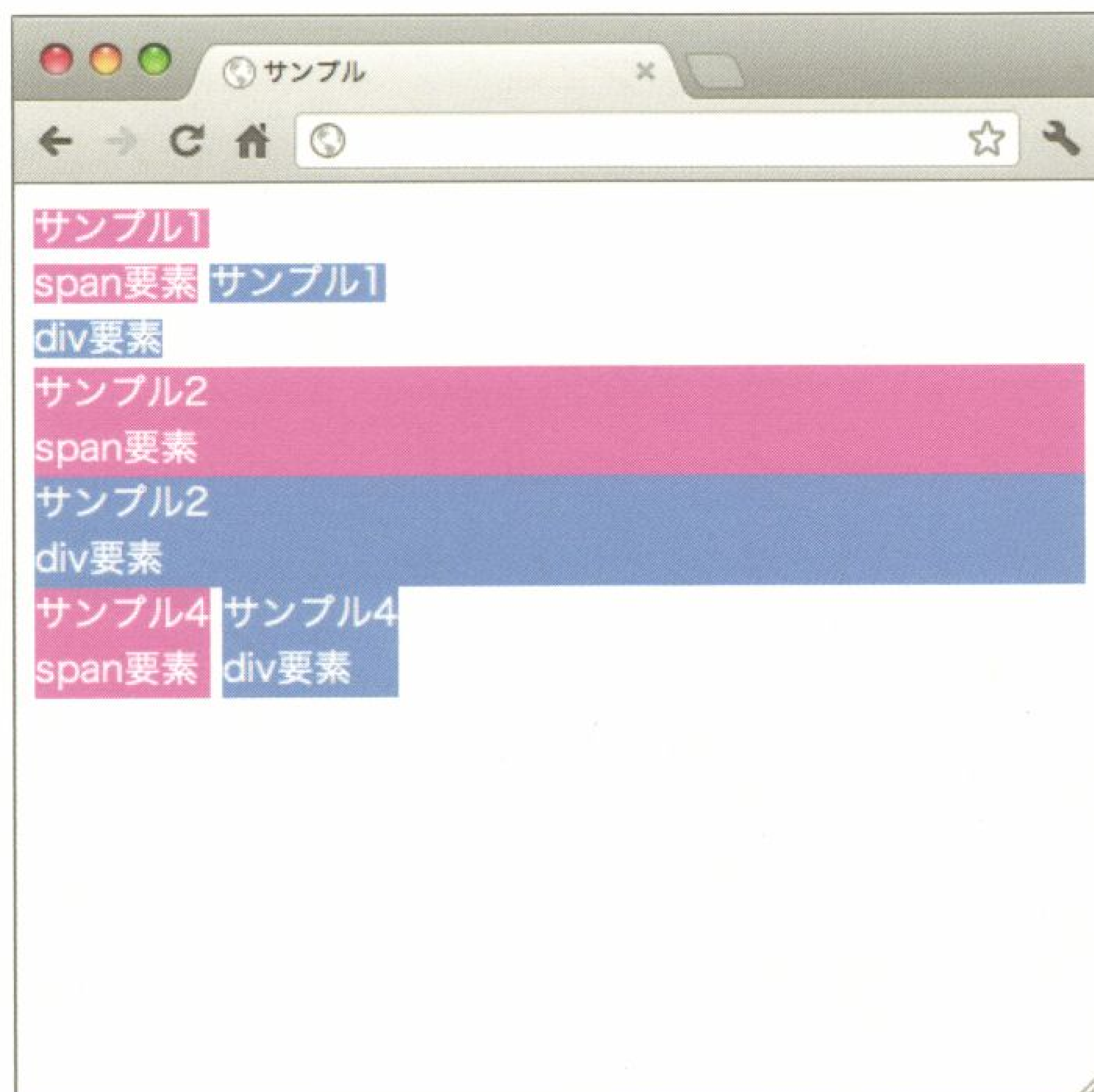
同様に横に並ぶけれどもその内部がブロックレベル要素のようになっているのが「inline-block」です。

CSS　sample/chapter-08/lecture-8-3/01-display.css

```
span, div { color: #fff; }
span { background: #f6d; }
div { background: #69e; }

.sample1 { display: inline; }
.sample2 { display: block; }
.sample3 { display: none; }
.sample4 { display: inline-block; }
```

それぞれにdisplayプロパティの異なる値を指定



CSS適用後の表示

見えない状態にする

displayプロパティの値として「none」を指定すると、その要素のボックスは消えてなくなりました。**visibilityプロパティ**を使用すると、要素自体を消すのではなく、見えないのだけれども場所は確保されている状態(つまりあたかも透明になったような状態)にすることができます。

visibilityに指定できる値

- `visible`
  ボックスを見える状態にします。

- `hidden`
  ボックスを見えない状態にします。

visibilityプロパティの使用例

では、具体的にどうなるのかを、2つの画像を表示させたサンプルで確認してみましょう。HTMLソースは次の通りです。

HTML　　sample/chapter-08/lecture-8-3/02.html

```
<img id="sample1" src="tpp.jpg" alt="タンポポの画像">
<img id="sample2" src="tpp.jpg" alt="タンポポの画像">
```

サンプルのHTMLソース。img要素が2つあるのみ

はじめに、この状態での表示を確認しておきましょう。この段階では単純に画像が2つ並んで表示されています。



CSSを適用していない状態での表示

では、次のCSSを適用させてみましょう。
左側の画像の画像は消えましたが、右側の画像は表示位置が変わることもなく、そのまま表示されています。

CSS　sample/chapter-08/lecture-8-3/02-visibility.css

```
#sample1 { visibility: hidden; }
```

左側の画像に「visibility: hidden;」を指定



左側の画像は表示されなくなったが、右側の画像の表示位置などは一切変わっていない

はみ出る部分の表示方法を設定

overflowプロパティを使用すると、ボックスの幅や高さを指定している状態で、要素内容が**ボックス内に入りきらなくなってしまった**ときにどのように表示させるのかを設定することができます。ただし、このプロパティを適用できるのはブロックレベルの状態の要素に限ります。次の値が指定できます。

overflowに指定できる値

- visible
  ボックスからはみ出た部分も表示します。
- hidden
  ボックスからはみ出た部分は表示しません。
- scroll
  ボックスからはみ出た部分は表示しませんが、スクロールによってすべての内容が見られるようにします。
- auto
  必要に応じて(内容が入りきらなくなると)スクロール可能にします。

次のサンプルでは、幅100ピクセル・高さ100ピクセルのボックス中に、幅200ピクセル・高さ200ピクセルの画像を表示させています。overflowプロパティの値によって、次のように表示結果が変わります。

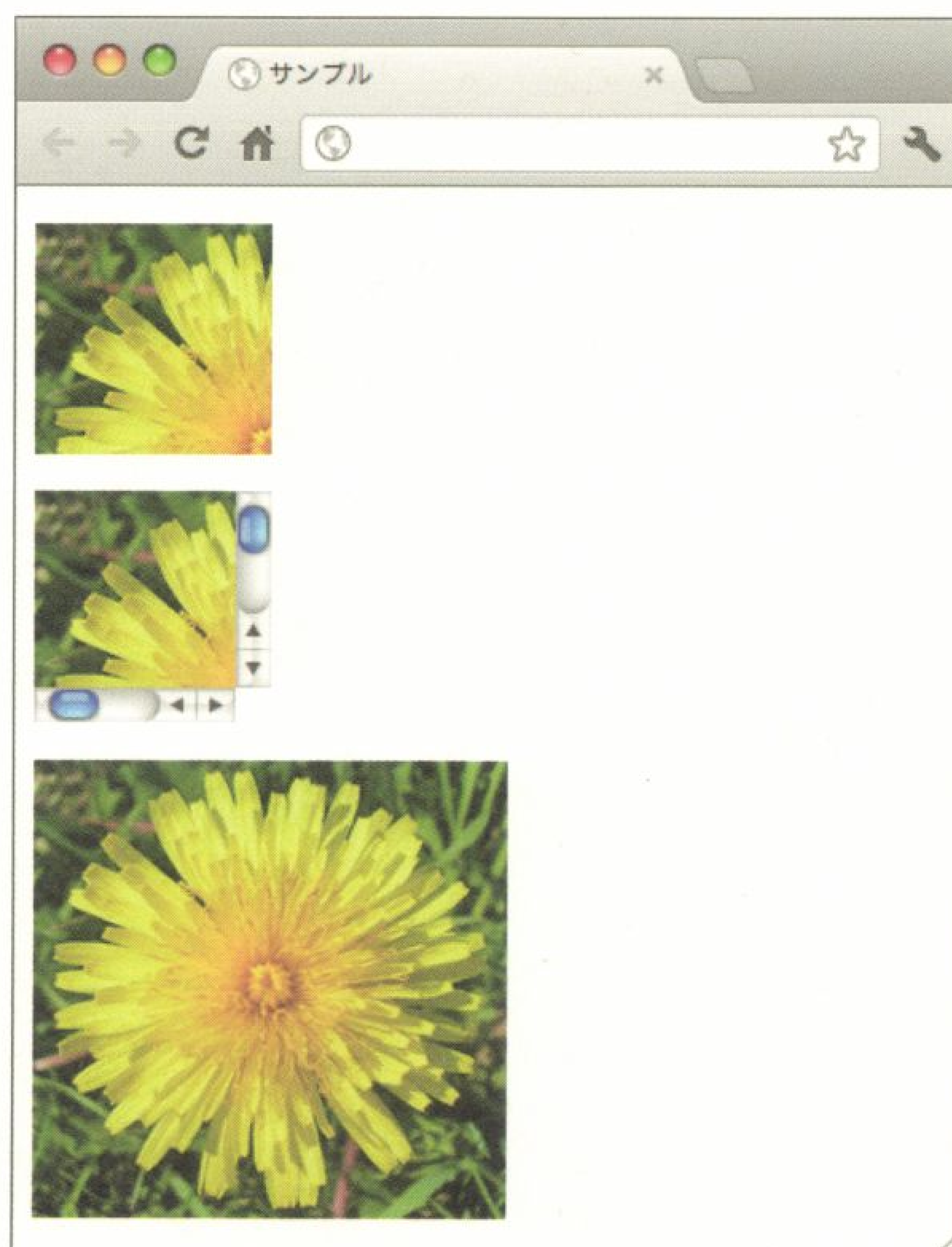
HTML sample/chapter-08/lecture-8-3/03.html

```
<p id="sample1">
<img src="tpp.jpg" alt="タンポポの画像">
</p>
<p id="sample2">
<img src="tpp.jpg" alt="タンポポの画像">
</p>
<p id="sample3">
<img src="tpp.jpg" alt="タンポポの画像">
</p>
```

CSS sample/chapter-08/lecture-8-3/03-overflow.css

```
p {
  width: 100px;
  height: 100px;
}
#sample1 { overflow: hidden; }
#sample2 { overflow: scroll; }
#sample3 { overflow: visible; }
```

overflowプロパティの指定例



上のソースコードの表示結果（200×200ピクセルの画像を100×100ピクセルのボックス内に「hidden」「scroll」「visible」の順で表示させた状態）

LECTURE 8-4

ナビゲーションの作り方

では、この章で覚えた要素とプロパティをどのように使ってナビゲーションを作るのかを、簡単な例で紹介します。まずは、HTML側のマークアップから確認していきましょう。

ナビゲーションのマークアップ

ここでは、ヘッダー部分によくあるようなグローバル・ナビゲーションを作成すると仮定して、全体をnav要素の中に入れます。その内部にはul要素を配置して、ナビゲーションの各項目はliの中に入れます。
各項目はナビゲーションとして機能するようにa要素でリンクにしておきます。

HTML

```
<nav>
  <ul>
    <li><a href="a.html">項目A</a></li>
    <li><a href="b.html">項目B</a></li>
    <li><a href="c.html">項目C</a></li>
    <li><a href="d.html">項目D</a></li>
    <li><a href="e.html">項目E</a></li>
  </ul>
</nav>
```

ナビゲーション部分のマークアップ



この時点では、普通のリストとして表示される

CHAPTER 8

リストの項目を横に並べる

ここから徐々にCSSを追加して、ナビゲーションらしく見えるようにしていきます。まずはリストの各項目が横に並ぶようにします。リストには行頭記号がついていたりとそのままでは扱いにくいので、li要素に「display: block;」を指定します。
こうすることで行頭記号も消えて、li要素が普通のブロックレベル要素と同じように扱えるようになるわけです。あとは、Chapter 7で学習した段組みの要領でfloatプロパティを指定すると各項目は横に並びます。ちなみに、body要素にマージンを指定しているのは、サンプルとして見やすくするためです。

CSS

```
body {
  margin-top: 80px;
}

nav li {
  display: block;
  width: 80px;
  float: left;
}
```

リストの各項目を横に並べるためのソースコード



行頭記号が消え、リストの項目が横に並んだ

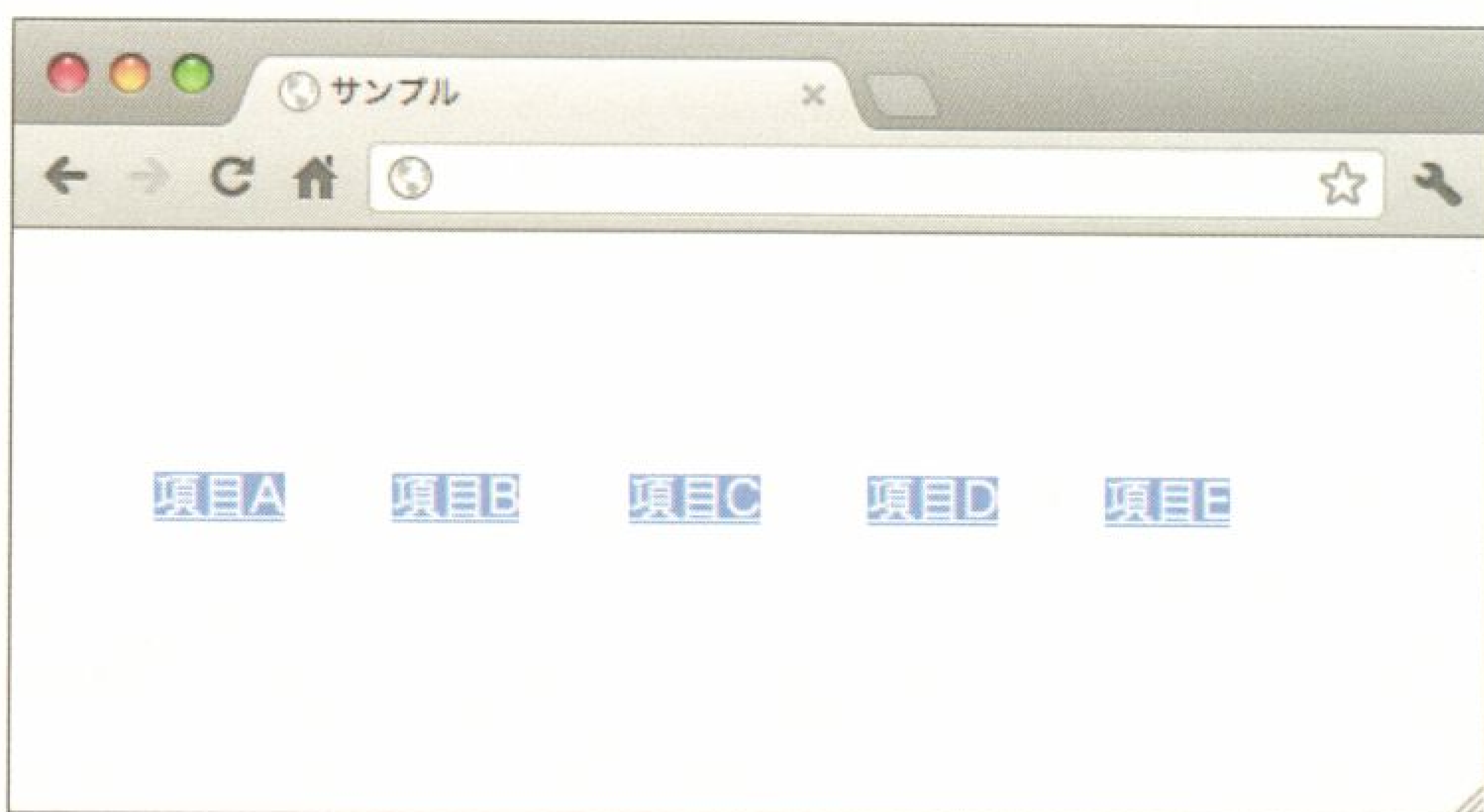
リンクの範囲を確認する

この状態だとボックスの範囲などが分かりにくいので、各項目内のa要素に背景色（と文字色）を指定します。
背景色を表示させてみると、背景がつくのはテキストのある狭い範囲に限定されていることが分かります。リンクとしてクリックできるのも背景が表示されている範囲ですので、その領域を広げる必要があります。

CSS

```
nav a {
  color: #fff;
  background: #69e;
}
```

a要素に背景色と文字色を指定する



この時点でクリックが可能なのは、水色の背景が表示されている狭い領域のみ

リンクの範囲を拡張する

a要素はインライン要素であるため、widthプロパティやheightプロパティを指定して大きさを変更することはできません。インライン要素の場合は、高さはフォントサイズや行間などで決まりますし、状況によって行を折り返す場合もあるので幅も指定できないのです。それではナビゲーションの項目としては不都合ですので、a要素もブロックレベル化してしまいましょう。
次の指定を追加します。

CHAPTER 8

CSS

```
nav a {
  color: #fff;
  background: #69e;
  display: block;  ← この行を追加
}
```

a要素もブロックレベル化する



背景が表示される範囲(=クリック可能な範囲)が広がった

表示を調整する

この状態ではナビゲーションらしく見えないので、表示を調整します。まず、行間とパディングの指定を追加して背景の表示される領域を調整し、各項目の左側に白いボーダーを表示させて境界がハッキリと分かるようにします。あとはテキストを中央揃えにして下線を消したら、シンプルではありますがナビゲーションらしくなりました。

CSS

```
nav a {
  color: #fff;
  background: #69e;
  display: block;
  line-height: 1.0;
  padding: 6px 0;
  border-left: 1px solid #fff;       追加
  text-align: center;
  text-decoration: none;
}
```

ナビゲーションらしく見えるように表示を調整する



ナビゲーションらしく見えるようになった

カーソルが上にあるときの処理

一般に、ナビゲーションの項目はカーソルをのせると背景が変わるなどなんらかの変化を見せます。最後にその処理を追加しましょう(これで完成ですのでナビゲーション部分全体の表示指定を掲載しておきます)。ナビゲーション作成のポイントは、li要素とa要素の両方に「display: block;」を指定することです。あとは段組みの要領で横に並べたり、細かい表示の調整をおこなうだけで簡単に作成できます。

HTML　sample/chapter-08/lecture-8-4/01.html

```
<nav>
  <ul>
    <li><a href="a.html">項目A</a></li>
    <li><a href="b.html">項目B</a></li>
    <li><a href="c.html">項目C</a></li>
    <li><a href="d.html">項目D</a></li>
    <li><a href="e.html">項目E</a></li>
  </ul>
</nav>
```

CSS　sample/chapter-08/lecture-8-4/01-navigation.css

```
nav li {
  display: block;
  width: 80px;
  float: left;
}
nav a {
  color: #fff;
  background: #69e;
  display: block;
```

```
  line-height: 1.0;
  padding: 6px 0;
  border-left: 1px solid #fff;
  text-align: center;
  text-decoration: none;
}
nav a:hover {
  background: #f6d;
}
```

追加

カーソルをのせたときの処理を追加



カーソルをのせると背景色が変わるようになった

CHAPTER 9

フォームとテーブル

HTMLでは、検索などに使用するテキスト入力欄のような入力・選択のためのパーツが多く用意されています。Chapter 9では、そのようなパーツの扱い方とテーブル（表）のマークアップの仕方、およびそれらに関連するCSSのプロパティについて説明します。

LECTURE 9-1

フォーム関連の要素

HTML5では、フォームに関連する新しい要素が追加されているだけでなく、既存の要素に対しても新しい属性が大量に追加されています。しかし、Operaのようにそれらの機能の多くをサポートしているブラウザもある一方で、現時点ではそれらをサポートしていないブラウザも多く存在しています。そのため、(特に確実にデータを送信する必要のあるフォームにおいては)そのような新機能の多くはまだあまり利用されていないのが現状です。というわけで、ここでは現時点で利用できる要素と属性を中心に、フォームについて説明していきます。

フォーム全体を囲む要素

form要素は、要素内として含んでいるフォーム関連要素で入力・選択した**データの送信先**や**送信方法**を指定する要素です。テキスト入力欄やメニューなどは必ずしもこの要素の中に入れて使用する必要はありませんが、データをサーバーなどに送信するのであればform要素内に入れる必要があります※14。form要素には次の属性が指定できます。

form要素に指定できる属性

- **action="送信先のURL"**
  データの送信先のURLを指定します。

- **method="送信方法"**
  データの送信方法を指定します。「get(初期値)」または「post」が指定できます。「get」はURLの後ろにデータをつけ加えて送信する方法です。「post」はURLとは別に(HTTPリクエストとは別のデータ本体として)データを送信します。

- **enctype="MIMEタイプ"**
  method属性の値が「post」のときの、データを送信する際のMIMEタイプを指定します。「application/x-www-form-urlencoded(初期値)」「multipart/form-data」「text/plain」のいずれかが指定できます。データとしてファイルを送信する場合には「multipart/form-data」を指定する必要があります。

- **name="フォームの名前"**
  フォームを参照するための名前を指定します。

※14:HTML5では、form要素の中に含まれていない要素でもサーバーに送信することが可能なように新しい属性が追加されていますが、ブラウザの対応状況から現時点ではほとんど利用されていないようです。

入力欄やボタンを生成する要素 【HTML5改】

フォームで使用されるテキスト入力欄や各種ボタン類の多くは、input要素というたった一種類の空要素によって生成されます。input要素をどのフォーム部品にするのかは、type属性で指定します。フォーム部品の種類によって、使用する属性とその機能は違ってきます。ユーザーによって入力・選択された値は、name属性で指定した名前とペアで送信されます。

input要素に指定できる属性

・type="フォーム部品の種類"
このinput要素を、どのフォーム部品にするのかを次のキーワードで指定します。

キーワード	フォーム部品の種類
text	1行のテキスト入力欄(一般テキスト用)
password	1行のテキスト入力欄(一般パスワード用)
checkbox	チェックボックス
radio	ラジオボタン
file	ファイル送信用部品
hidden	画面上には表示させずに送信するテキスト
submit	送信ボタン
reset	リセットボタン
button	汎用ボタン
image	画像の送信ボタン

・name="部品の名前"
このフォーム部品の名前を指定します。入力・選択されたデータは、この名前とペアで送信されます。同じ選択項目内でのチェックボックスまたはラジオボタンには同じ名前をつける必要があります。

・value="初期値／ラベル／送信値"
テキスト入力欄の場合は、そこに最初から入力されている初期値となります。ボタンの場合は、そのボタンのラベルとなります。チェックボックスまたはラジオボタンの場合は、その項目を選択したときにサーバーに送信される値となります。

・size="文字数"
テキスト入力欄の文字数を指定します。この属性に指定した値によってテキスト入力欄の幅が変化します。初期値は20です。

・maxlength="最大文字数"
テキスト入力欄に入力できる最大の文字数を指定します。

・checked
チェックボックスまたはラジオボタンを選択した状態にします。

input要素に指定できる属性（続き）

・readonly
　このフォーム部品を変更不可（選択は可能）の状態にします。

・disabled
　このフォーム部品を変更・選択不可の状態にします。

・src="画像のURL"
　画像の送信ボタンにする際の「画像のURL」を指定します。

・width="幅"
　画像の送信ボタンの画像の幅（実際の幅ではなく表示させる幅）をピクセル数で指定します。

・height="高さ"
　画像の送信ボタンの画像の高さ（実際の高さではなく表示させる高さ）をピクセル数で指定します。

・alt="代替テキスト"
　画像の送信ボタンの画像が表示できない場合に、その代わりとして使用するテキストを指定します。

▶▶ input要素の使用例

では、input要素が実際にどのように使用され、ブラウザではどのように表示されるのかをサンプルで確認してみましょう。

HTML　　sample/chapter-09/lecture-9-1/01.html

```
<form action="sample.cgi" method="post">
<p>
text：<input type="text" name="type01">
</p>
<p>
password：<input type="password" name="type02">
</p>
<p>
checkbox：
<input type="checkbox" name="type03" value="c1" checked>項目1
<input type="checkbox" name="type03" value="c2">項目2
<input type="checkbox" name="type03" value="c3">項目3
</p>
<p>
radio：
<input type="radio" name="type04" value="r1" checked>項目1
<input type="radio" name="type04" value="r2">項目2
<input type="radio" name="type04" value="r3">項目3
```

```
</p>
<p>
file：<input type="file" name="type05">
</p>
<p>
hidden：<input type="hidden" value="h1" name="type06">
</p>
<p>
submit：<input type="submit">
</p>
<p>
reset：<input type="reset">
</p>
<p>
button：<input type="button" value="ボタン">
</p>
<p>
image：<input type="image" src="shell.jpg" alt="送信">
</p>
</form>
```

input要素の使用例。これらのフォーム部品はすべてinput要素のtype属性に異なる値を指定したもの

input要素の表示例。表示結果は、ブラウザやOSの種類によって異なる

COLUMN

仕様上はもっと多くの部品が用意されている!?

実はここまでに紹介したフォーム部品はすべてHTML4ですでに定義されていたもので、HTML5ではさらに右のようなキーワードがtype属性の値として指定できるようになっています。ただし、それらの多くに対応しているのは現時点ではOperaぐらいで、Internet Explorer 9などはすべてに未対応となっています。これらは仕様がまだ確定しているものでもありませんし、一般的なページで利用できるようになるにはまだまだ時間がかかると思われます。

キーワード	フォーム部品の種類
search	1行のテキスト入力欄(検索用)
tel	1行のテキスト入力欄(電話番号用)
url	1行のテキスト入力欄(URL用)
email	1行のテキスト入力欄(メールアドレス用)
datetime	日時の入力用部品(タイムゾーンあり)
datetime-local	日時の入力用部品(タイムゾーンなし)
date	年月日入力用部品
month	年月入力用部品
week	年週入力用部品
time	時刻入力用部品
number	数値入力用部品
range	スライド型部品
color	色選択用部品

HTML5で追加されたinput要素のtype属性の値

sample/chapter-09/lecture-9-1/02.html

```
<p>search：<input type="search"></p>
<p>tel：<input type="tel"></p>
<p>url：<input type="url"></p>
<p>email：<input type="email"></p>
<p>datetime：<input type="datetime"></p>
<p>datetime-local：<input type="datetime-local"></p>
<p>date：<input type="date"></p>
<p>month：<input type="month"></p>
<p>week：<input type="week"></p>
<p>time：<input type="time"></p>
<p>number：<input type="number"></p>
<p>range：<input type="range"></p>
<p>color：<input type="color"></p>
```

HTML5で追加されたtype属性の値の使用例

左のサンプルソースをOpera 11で表示させたところ

複数行のテキスト用の入力欄

textarea要素は、**複数行のテキスト入力欄**を表示させる要素です。input要素で生成可能なのは1行のテキスト入力欄だけで、複数行のテキスト入力欄が必要な場合にはtextarea要素を使用する必要があります。この要素は空要素ではなく、要素内容として入れたテキストが、初期状態でテキスト入力欄に入力された状態で表示されます。
次の属性が指定できます。

textarea要素に指定できる属性

- **cols="文字数"**
  テキスト入力欄の1行の文字数を指定します。この属性に指定した値によってテキスト入力欄の幅が変化します。初期値は20です。

- **rows="行数"**
  テキスト入力欄の行数を指定します。この属性に指定した値によってテキスト入力欄の高さが変化します。初期値は2です。

- **name="部品の名前"**
  このフォーム部品の名前を指定します。入力されたデータは、この名前とペアで送信されます。

- **maxlength="最大文字数"**
  入力可能な最大の文字数を指定します。

- **readonly**
  このフォーム部品を変更不可(選択は可能)の状態にします。

- **disabled**
  このフォーム部品を変更・選択不可の状態にします。

CHAPTER 9

HTML　sample/chapter-09/lecture-9-1/03.html

```
<form action="sample.cgi" method="post">
<p>
<textarea rows="7" cols="50">
サンプルテキスト
</textarea>
</p>
</form>
```

textarea要素の使用例



前ページのソースコードの表示

要素内容をラベルとして表示する要素

基本的なボタンはinput要素で生成可能なのですが、それとは別に**button要素**というボタン専用の要素も用意されています。input要素は空要素でしたが、button要素の場合は要素内容をそのままボタン上のラベルとして表示できる点が大きく異なります。

button要素に指定できる属性

- **type="ボタンの種類"**
  ボタンの種類を次のキーワードで指定します。

キーワード	ボタンの種類
submit	送信ボタン(初期値)
reset	リセットボタン
button	汎用ボタン

- **name="部品の名前"**
  このフォーム部品の名前を指定します。この名前とvalue属性の値がペアで送信されます。

- **disabled**
  このフォーム部品を変更・選択不可の状態にします。

- **value="送信値"**
  サーバーに送信される値を指定します。この値とname属性の値がペアで送信されます。

HTML　sample/chapter-09/lecture-9-1/04.html

```
<p>
<button type="submit">送信ボタン</button>
<button type="reset">リセットボタン</button>
<button type="button">
<strong>汎用ボタン</strong>
<br>
<img src="shell.jpg">
</button>
</p>
```

CSS　sample/chapter-09/lecture-9-1/04-button.css

```
button[type="button"] {
  font-size: large;
  border-radius: 10px;
  color: white;
  background: orange;
}
```

button要素の使用例



上のソースコードの表示

メニューを構成する要素

メニューは、フォームの部品の中で唯一複数の要素の組み合わせで作られる部品です。メニューの構造は、リストと同じようになっていて、全体をselect要素で囲い、各選択肢はoption要素という空要素で指定します。それぞれに指定できる属性は次の通りです。

select要素に指定できる属性

- **name="部品の名前"**
  このフォーム部品の名前を指定します。選択されたデータは、この名前とペアで送信されます。
- **size="行数"**
  一度に見られる項目の数（行数）を指定します。この属性を指定すると、メニューではなくリストボックスになります。
- **multiple**
  複数の項目を選択できるようにします。
- **disabled**
  このフォーム部品を変更・選択不可の状態にします。

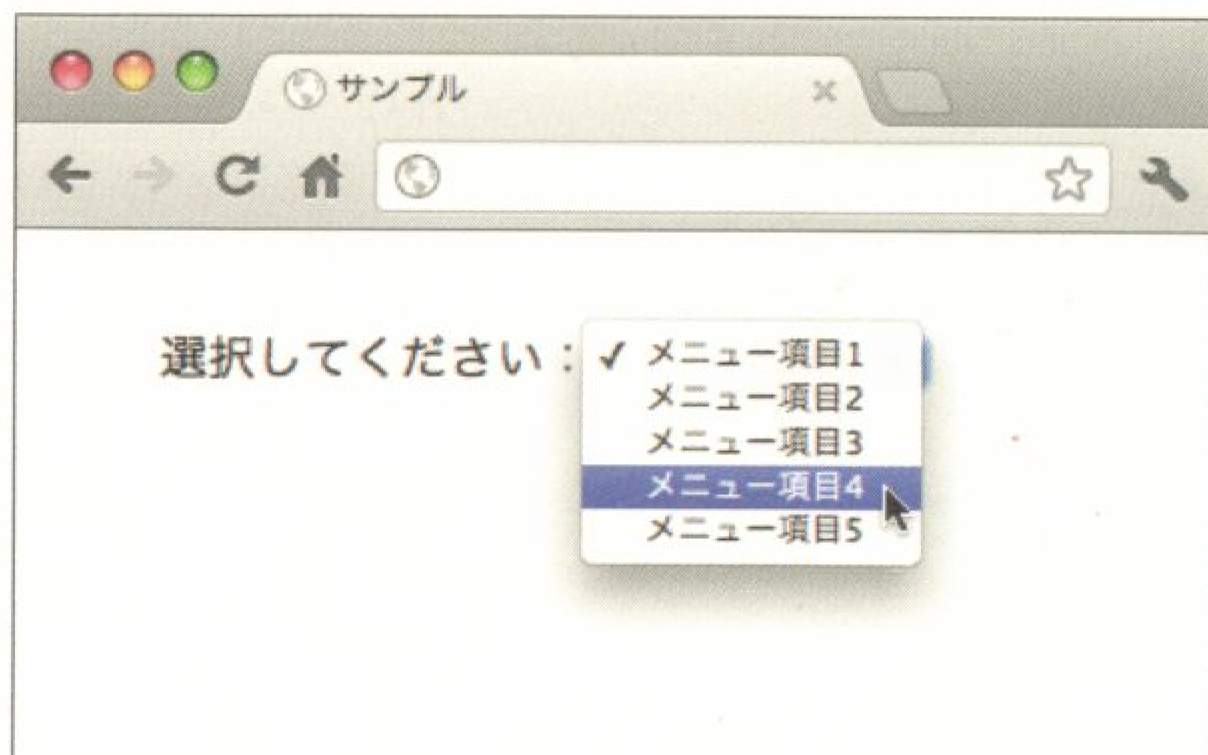
option要素に指定できる属性

- **value="送信値"**
  サーバーに送信される値を指定します。この値とselect要素のname属性の値がペアで送信されます。
- **selected**
  この項目を選択した状態にします。
- **disabled**
  このフォーム部品を変更・選択不可の状態にします。

HTML　sample/chapter-09/lecture-9-1/05.html

```
<p>
選択してください：
<select>
  <option>メニュー項目1</option>
  <option>メニュー項目2</option>
  <option>メニュー項目3</option>
  <option>メニュー項目4</option>
  <option>メニュー項目5</option>
</select>
</p>
```

select要素・option要素の使用例



前ページのソースコードの表示

フォーム部品とテキストを関連づける要素

たとえば「名前：」のようにテキスト入力欄の前にテキストが配置されていたとしても、そのテキストとテキスト入力欄は内部的には結びついていません。そのため、その「名前：」と書かれた部分をクリックしたとしても、テキスト入力欄には何の反応もありません（入力を開始するには直接テキスト入力欄をクリックする必要があります）。同様に、チェックボックスやラジオボタンもテキスト部分をクリックしても一切反応はなく、直接ボタン部分をクリックしなければなりません。しかし、それでは一般的なアプリケーションとは反応が違うことになり使いにくいので、テキストとフォーム部品を内部的に関連づけるための要素が用意されています。それが**label要素**です。

▶▶ label要素の使い方

label要素でテキストとフォーム部品を関連づけるには2つの方法があります。1つはテキストとフォーム部品を一緒にlabel要素の内容として入れてしまう方法です。もう1つは、フォーム部品にid属性を指定しておき、内容としてテキストだけを入れたlabel要素のfor属性の値にそのid属性の値を入れて関連づける方法です。ただし、Internet Explorer 6がfor属性を使った方法にしか対応していないため、一般的には現在でもfor属性を使う方法が多く採用されています。
label要素の内容として入れられるのは、インライン要素だけです（各種フォーム部品もインライン要素に分類されます）。ブロックレベル要素は入れられませんので注意してください。

label要素に指定できる属性

・**for="部品のid"**
　フォーム部品のid属性の値を指定して、このラベルと関連づけます。

HTML　sample/chapter-09/lecture-9-1/06.html

```
<p>
<label for="parts1">ラベル：</label>
<input id="parts1" type="text">
</p>
<p>
<input id="parts2" type="checkbox" name="cb">
<label for="parts2">項目A</label>
<input id="parts3" type="checkbox" name="cb">
<label for="parts3">項目B</label>
<input id="parts4" type="checkbox" name="cb">
<label for="parts4">項目C</label>
</p>
<p>
<input id="parts5" type="radio" name="rb">
<label for="parts5">項目D</label>
<input id="parts6" type="radio" name="rb">
<label for="parts6">項目E</label>
<input id="parts7" type="radio" name="rb">
<label for="parts7">項目F</label>
</p>
```

label要素の使用例

サンプル
ラベル：
項目A　項目B　項目C
項目D　項目E　項目F

上のソースコードの表示。テキスト部分をクリックすると反応するようになっている

フォーム部品などをグループ化する要素

各種フォーム部品やそのラベルのテキストなどをグループ化するには、**fieldset要素**を使用します。グループの表示方法が特に決められているわけではありませんが、一般的なブラウザではグループ全体を囲うような線が表示されます。

fieldset要素でグループ化した範囲にタイトルをつけるには、**legend要素**を使用します。legend要素は、fieldset要素でマークアップした範囲の先頭に1つだけ入れることができます。

fieldset要素に指定できる属性

- **name="部品の名前"**
  この要素の名前を指定します。

- **disabled**
  このフォーム部品を変更・選択不可の状態にします。

次の例では、前のサンプルをそのまま使ってfieldset要素でグループ化しています。

HTML　sample/chapter-09/lecture-9-1/07.html

```html
<fieldset>
<legend>グループのタイトル</legend>
<p>
<label for="parts1">ラベル：</label>
<input id="parts1" type="text">
</p>
<p>
<input id="parts2" type="checkbox" name="cb">
<label for="parts2">項目A</label>
<input id="parts3" type="checkbox" name="cb">
<label for="parts3">項目B</label>
<input id="parts4" type="checkbox" name="cb">
<label for="parts4">項目C</label>
</p>
<p>
<input id="parts5" type="radio" name="rb">
<label for="parts5">項目D</label>
<input id="parts6" type="radio" name="rb">
<label for="parts6">項目E</label>
<input id="parts7" type="radio" name="rb">
<label for="parts7">項目F</label>
</p>
</fieldset>
```

fieldset要素とのlegend要素使用例。legend要素を使用する場合は、必ずグループ内の先頭に入れる

上のソースコードの表示

LECTURE 9-2

フォームの関連のプロパティ

ここで、フォームのために用意されているというわけではありませんが、フォームと関連して使用されることが多いプロパティをいくつか紹介しておきましょう。

リサイズ可能にする 【CSS3新】

Google ChromeやSafari、Firefoxの比較的新しいバージョンを使用している人であれば、複数行のテキスト入力欄の大きさが変えられるようになっていることに気づいていると思います。適用対象がテキスト入力欄に限定されているわけではありませんが、このようにボックスの大きさをユーザーが変更できるようにするかどうかは**resizeプロパティ**で設定できます。

resizeに指定できる値

- **both**
  幅と高さの両方をリサイズ可能にします。

- **horizontal**
  幅だけをリサイズ可能にします。

- **vertical**
  高さだけをリサイズ可能にします。

- **none**
  リサイズができない状態にします。

resizeプロパティの初期値は「none」なのですが、Google ChromeやSafari、Firefoxではデフォルトスタイルシートでtextarea要素のresizeプロパティの値が「both」に設定されています。そのため、他の値を指定しないかぎり、幅と高さの両方がリサイズ可能な状態となっていますので注意してください。

▶▶ ブラウザの対応状況について

resizeはCSS3のプロパティですが、すでに勧告候補となっていますのでベンダープレフィックスは特につける必要はありません。このプロパティにはFirefox 4以降のほか、Google ChromeとSafariが古いバージョンから対応済みです。ただし、Internet Explorerはバージョン9でも未対応です。

HTML　sample/chapter-09/lecture-9-2/01.html

```
<p>
<textarea id="sample1" rows="3" cols="30">
初期状態</textarea>
</p>

<p>
<textarea id="sample2" rows="3" cols="30">
both(幅と高さを変更可)</textarea>
</p>

<p>
<textarea id="sample3" rows="3" cols="30">
horizontal(幅のみ変更可)</textarea>
</p>

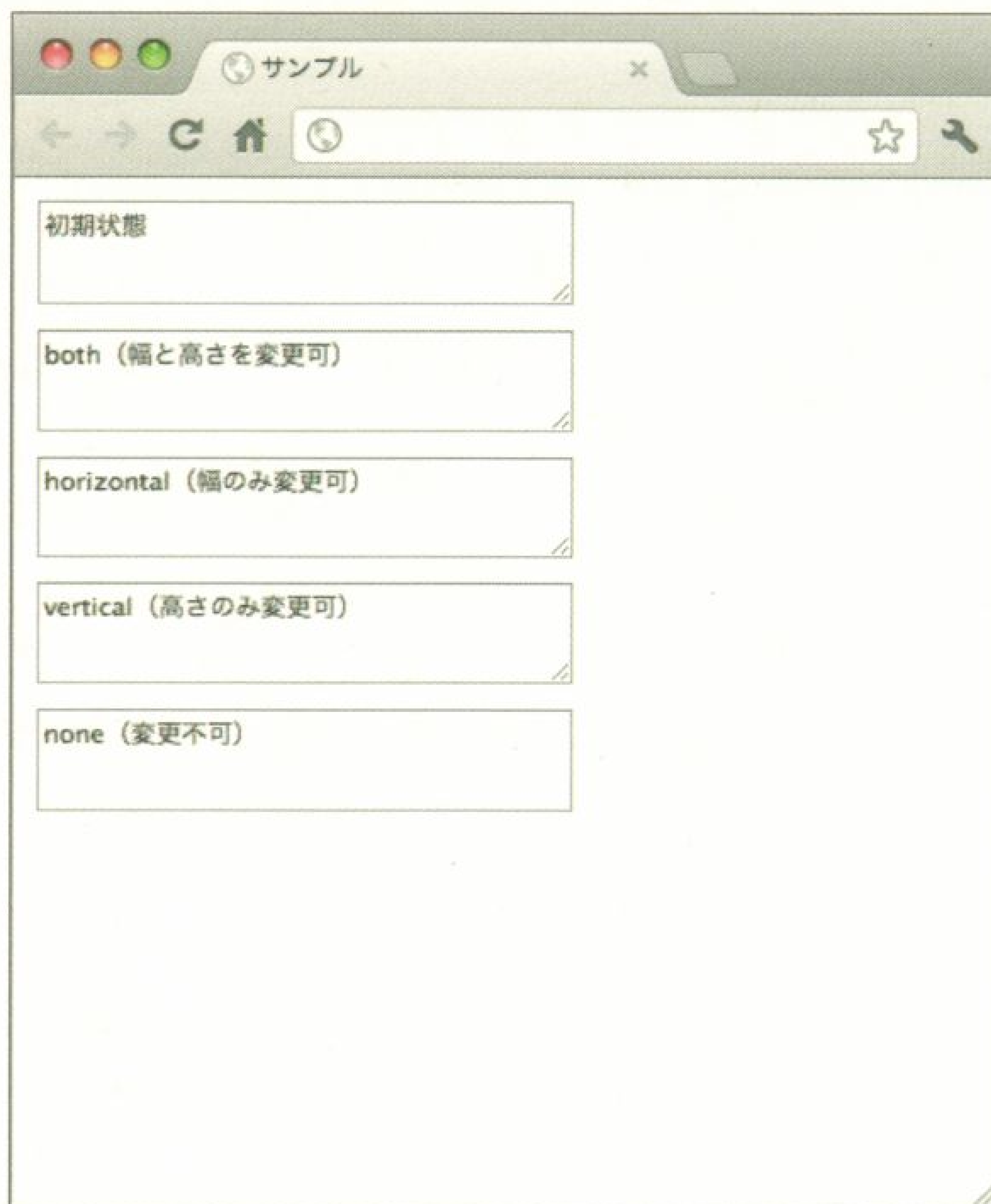
<p>
<textarea id="sample4" rows="3" cols="30">
vertical(高さのみ変更可)</textarea>
</p>

<p>
<textarea id="sample5" rows="3" cols="30">
none(変更不可)</textarea>
</p>
```

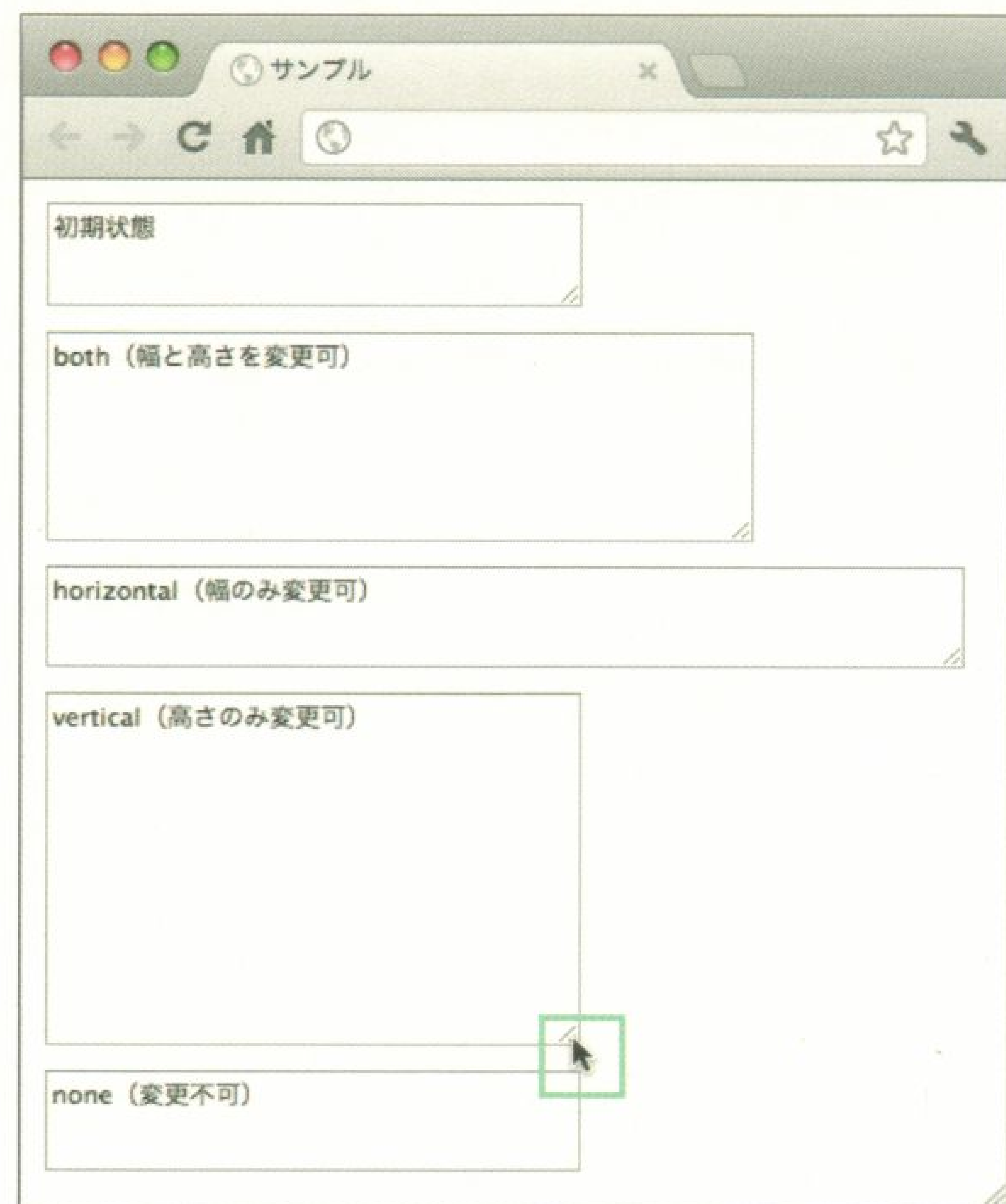
CSS　sample/chapter-09/lecture-9-2/01-resize.css

```
#sample2 { resize: both; }
#sample3 { resize: horizontal; }
#sample4 { resize: vertical; }
#sample5 { resize: none; }
```

resizeプロパティをtextarea要素に適用した例



前ページのソースコードを表示させたところ（何も操作していない状態）



このようにtextarea要素の大きさを変えることができる

ボックスに影をつける CSS3新

box-shadowプロパティを使用すると、text-shadowプロパティと同様の指定方法でボックスに影をつけることができます。ただし、box-shadowプロパティでは4つめの数値として影を拡張させる距離が指定でき、「inset」というキーワードを指定することで影をボックスの内部に表示させられるようにもなっています。次の値が指定できます。

box-shadowに指定できる値

- **inset**
  影をボックスの内側に表示させます。この値を指定しなければ、影はボックスの外側に表示されます。

- **単位つきの実数**
  影の「右方向への移動距離」「下方向への移動距離」「ぼかす範囲」「上下左右に拡張させる距離」は単位つきの実数で指定します。

- **色**
  色の書式に従って影の色を指定できます。

box-shadowに指定できる値（続き）

- **none**
  影を表示させません。

数値は、影の「右方向への移動距離」「下方向への移動距離」「ぼかす範囲」「上下左右に拡張させる距離」の順に半角スペースで区切って指定します。「ぼかす範囲」と「上下左右に拡張させる距離」は指定しなくてもかまいません。影の色は、これらの数値全体の前か後ろに半角スペースで区切って指定します。キーワード「inset」を指定する場合は、さらにそれら全体の前か後ろに半角スペースで区切って指定します。

▶▶ ブラウザの対応状況について

このプロパティには、Inernet Explorerもバージョン9から対応しており、ベンダープレフィックスなしで指定できます。Firefoxの古いバージョンやSafariにも対応させる場合にはそれらのベンダープレフィックスをつけた指定も追加してください。

HTML　sample/chapter-09/lecture-9-2/02.html

```html
<p>
<textarea id="sample1" rows="5" cols="30">
</textarea>
</p>

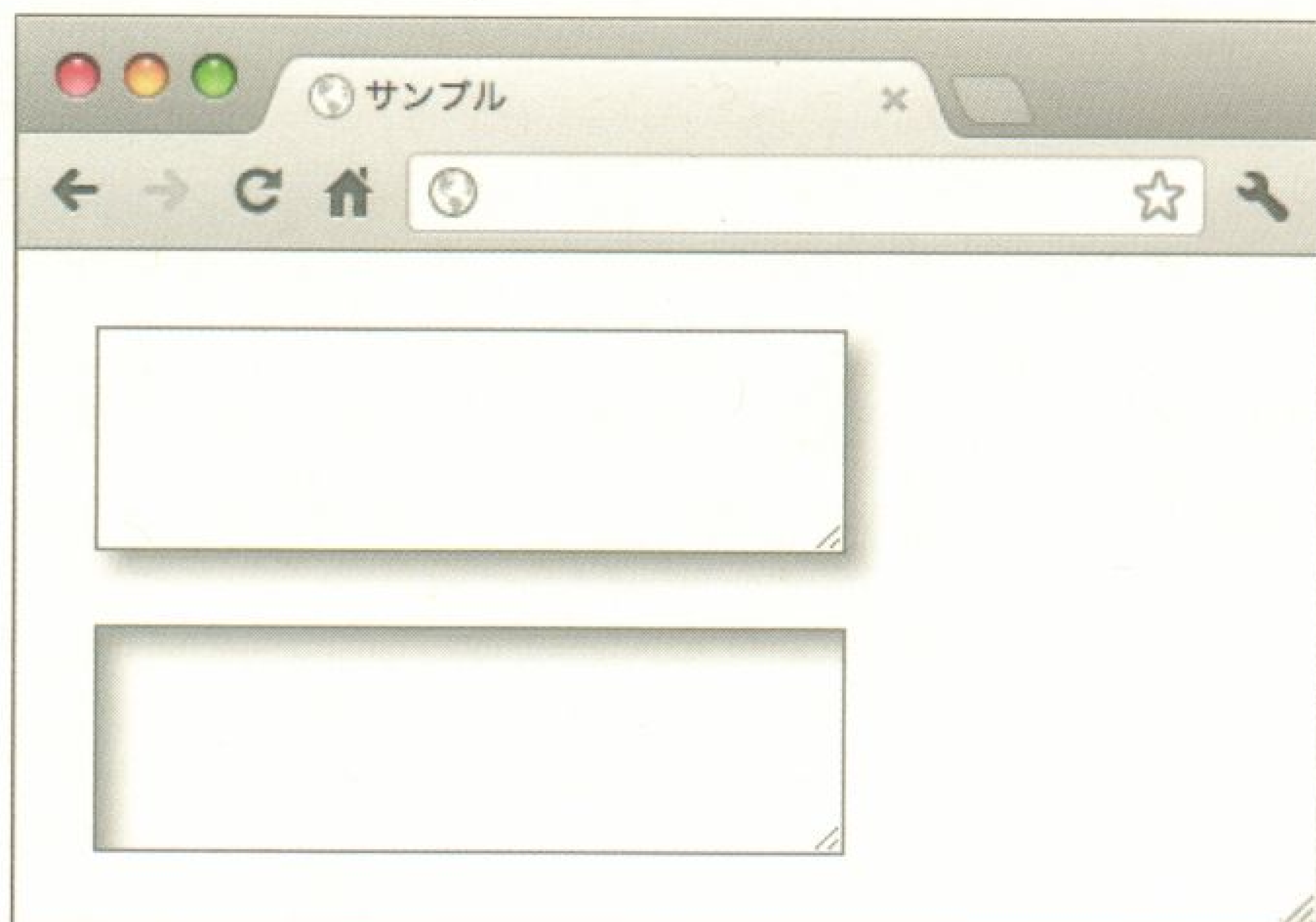
<p>
<textarea id="sample2" rows="5" cols="30">
</textarea>
</p>
```

CSS　sample/chapter-09/lecture-9-2/02-box-shadow.css

```css
textarea { border: 1px solid #666; }

#sample1 {
  -webkit-box-shadow: 5px 5px 10px #999;
  -moz-box-shadow: 5px 5px 10px #999;
  box-shadow: 5px 5px 10px #999;
}
#sample2 {
  -webkit-box-shadow: inset 5px 5px 10px #999;
  -moz-box-shadow: inset 5px 5px 10px #999;
  box-shadow: inset 5px 5px 10px #999;
}
```

box-shadowプロパティの使用例



前ページのソースコードの表示

COLUMN

textarea要素にbox-shadowプロパティが適用されない現象

実は、上のサンプルでtextarea要素にborderプロパティを指定しているのには訳があります。この指定がないと、Google ChromeとSafariでは、textarea要素にbox-shadowプロパティの影が表示されないのです。

Webkit系のブラウザ(Google ChromeやSafariなど)では、フォーム部品の表示をプラットフォームごとに統一したものとするためにwebkit-appearanceという独自のプロパティが設定されており、それによってこの現象が起こることが確認されています。このwebkit-appearanceによる表示設定は、textarea要素に「-webkit-appearance: none;」と指定するか、ボーダーを指定することで解除され、それによって影も表示されるようになります(これは2012年7月現在のGoogle ChromeとSafariで確認した情報で、将来的に変更される可能性があります)。

アウトライン

パソコンを操作していると、テキスト入力欄をクリックしたときやボタンの上にカーソルをのせたときなどに、そのまわりに線が表示されることがあります。それによって、たとえばテキスト入力欄がたくさん並んでいても、今キーボードを打つとどのテキスト入力欄に入力されるのかが一目で分かるようになっているわけです。そのような用途で使われている線が、アウトラインです。

アウトラインは、見た目はボーダーとよく似ていますが、ボックスの一部ではなくボックスの上に表示されます。そのため、アウトラインを表示させてもボックスの大きさが変わったり、レイアウトが崩れたりすることは一切ありません。表示される位置は、ボックスのボーダーのまわり（外側）となります。また、ボーダーのように上下左右を別々に指定することはできず、上下左右の線種・太さ・色は常に同じになります。

アウトライン関連としては次のプロパティが用意されています。各プロパティに指定できる値はボーダーとほぼ共通していますが、線種に「hidden」が指定できない点と、色に「invert」が指定できる点だけが異なっています。

プロパティ名	設定対象	指定できる値の数
outline-style	上下左右のアウトラインの線種	1
outline-width	上下左右のアウトラインの太さ	1
outline-color	上下左右のアウトラインの色	1
outline	上下左右のアウトラインの線種と太さと色	線種／太さ／色

アウトラインを設定するプロパティ。「線種／太さ／色」には半角スペースで区切って順不同で必要な値を指定できる。指定しない値は初期値となる

CHAPTER 9

outline-styleに指定できる値

- none
  アウトラインを表示しません。
- solid
  アウトラインの線種を実線にします。
- double
  アウトラインの線種を二重線にします。
- dotted
  アウトラインの線種を点線にします。

outline-styleに指定できる値（続き）

・**dashed**
アウトラインの線種を破線にします。

・**groove**
アウトラインの線自体が溝になっているようなボーダーにします。

・**ridge**
アウトラインの線自体が盛り上がっているようなボーダーにします。

・**inset**
アウトラインの内側の領域全体が低く見えるようなボーダーにします。

・**outset**
アウトラインの内側の領域全体が高く見えるようなボーダーにします。

outline-widthに指定できる値

・**単位つきの実数**
アウトラインの太さを単位つきの実数（5pxなど）で指定します。

・**thin, medium, thick**
「細い」「中くらい」「太い」という意味のキーワードで指定できます（実際に表示される太さはブラウザによって異なります）。

outline-colorに指定できる値

・**色**
色の書式に従って任意のアウトラインの色を指定します。

・**invert**
反転させた色にします。

outlineに指定できる値

・**outline-styleの値**
outline-styleに指定できる値が指定できます。

・**outline-widthの値**
outline-widthに指定できる値が指定できます。

・**outline-colorの値**
outline-colorに指定できる値が指定できます。

HTML　sample/chapter-09/lecture-9-2/03.html

```
<p>
<textarea rows="7" cols="40">
</textarea>
</p>
```

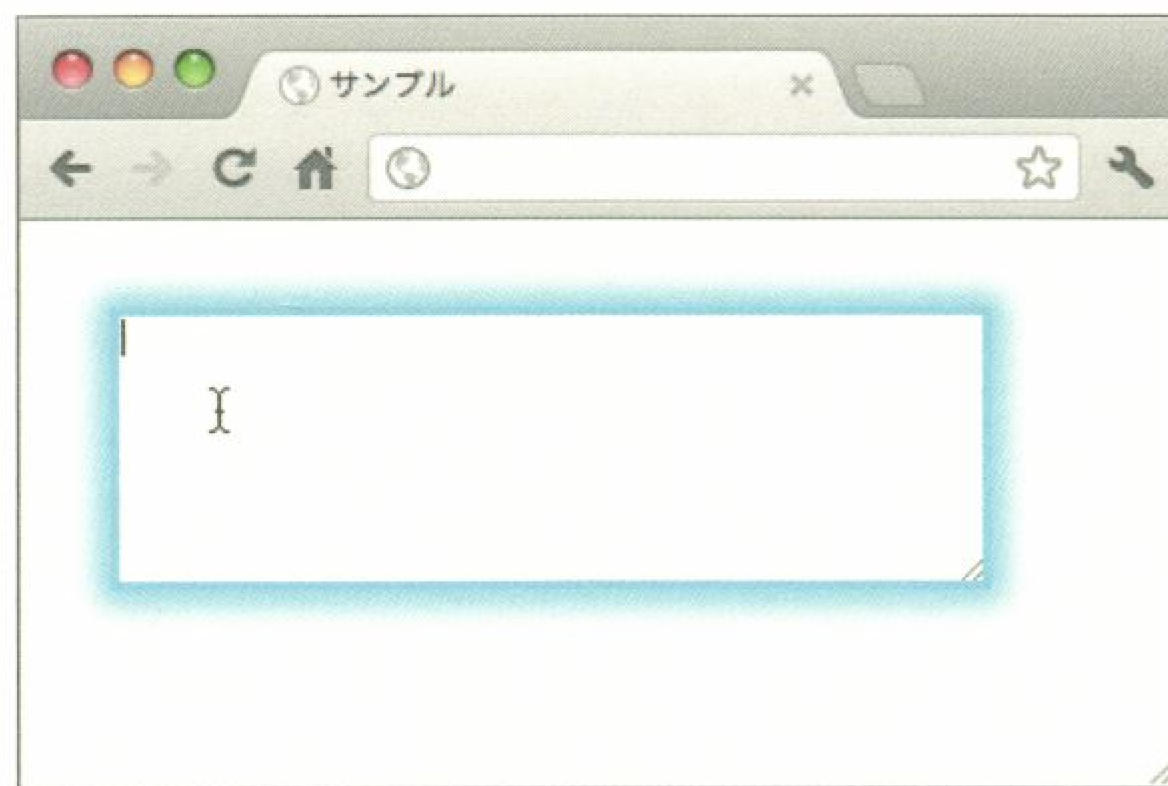
CSS　sample/chapter-09/lecture-9-2/03-outline.css

```
textarea {
  border: 1px solid #666;
}
textarea:focus {
  outline: 3px solid #6cf;
  box-shadow: 0 0 15px 5px #6cf;
}
```

outlineプロパティの使用例



上のソースコードを表示させたところ（何も操作していない状態）



テキスト入力欄の内部をクリックすると水色のアウトライン（と水色の影）が表示される

LECTURE 9-3

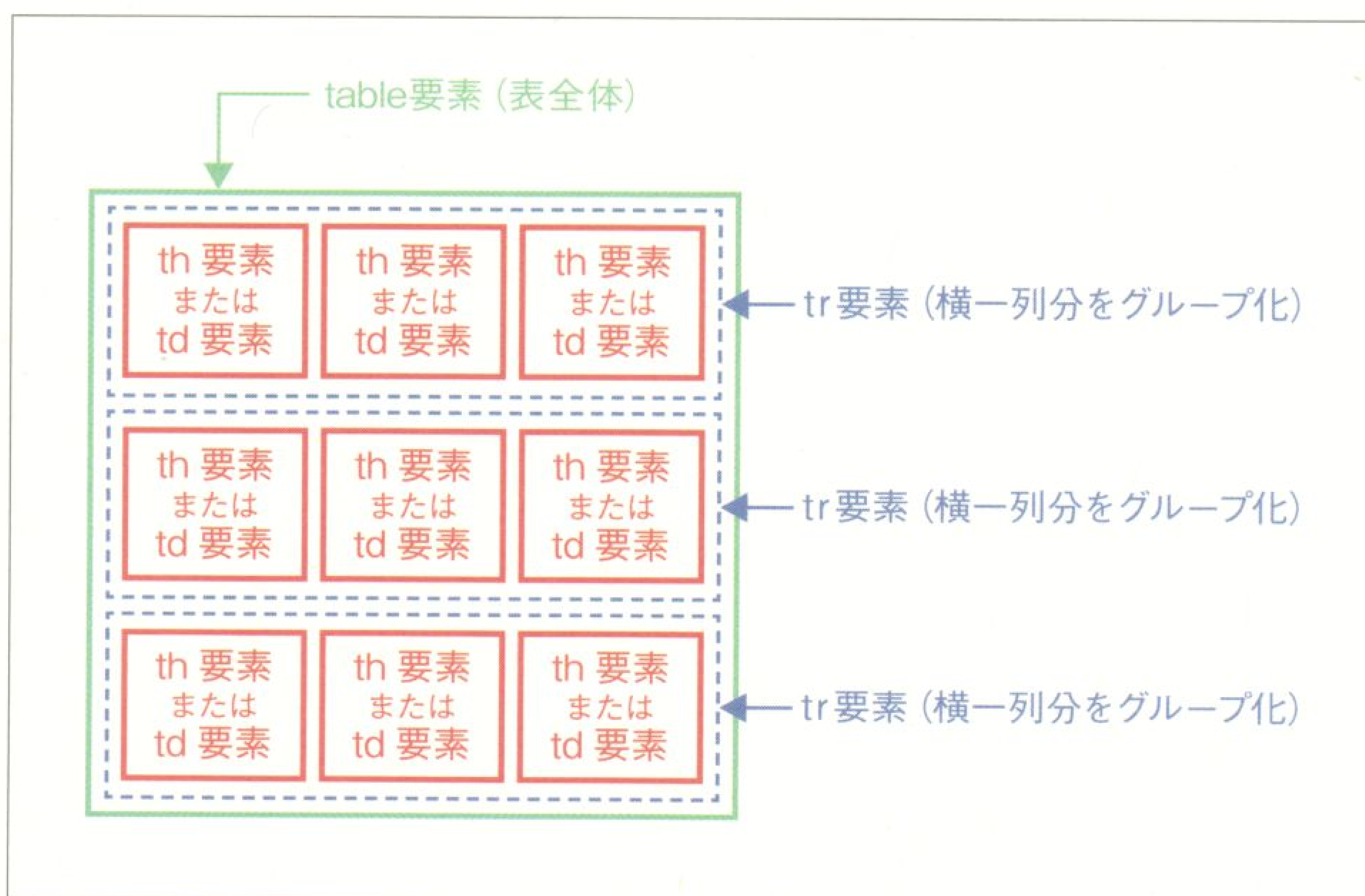
テーブル関連の要素

さて、ここからはテーブル（表）のマークアップの仕方について説明していきます。

テーブルを構成する要素

テーブルの構造は、HTMLの中ではもっとも複雑です。とはいえ、基本的な構造でいえば、リストの構造にさらにもう一種類の要素が加わっている程度のものです。まずはシンプルな基本構造から順に覚えていきましょう。

テーブルを作成するには、まずその全体を**table要素**のタグで囲います。その中にはテーブルのセルが入るのですが、その内容が縦列または横列の見出しであるセルは**th要素**（table header cell）としてマークアップし、その内容がデータであるセルは**td要素**（table data cell）としてマークアップします。そして、各セルは横一列ごとに**tr要素**（table row）でグループ化します。これがHTMLのテーブルの基本構造です。



HTMLのテーブルの基本構造

▶▶ table要素

table要素には、次の値が指定できます。border属性を指定しなければ、一般的なブラウザではテーブルのボーダーは表示されません(もちろんCSSを指定すれば表示されます)。

table要素に指定できる属性

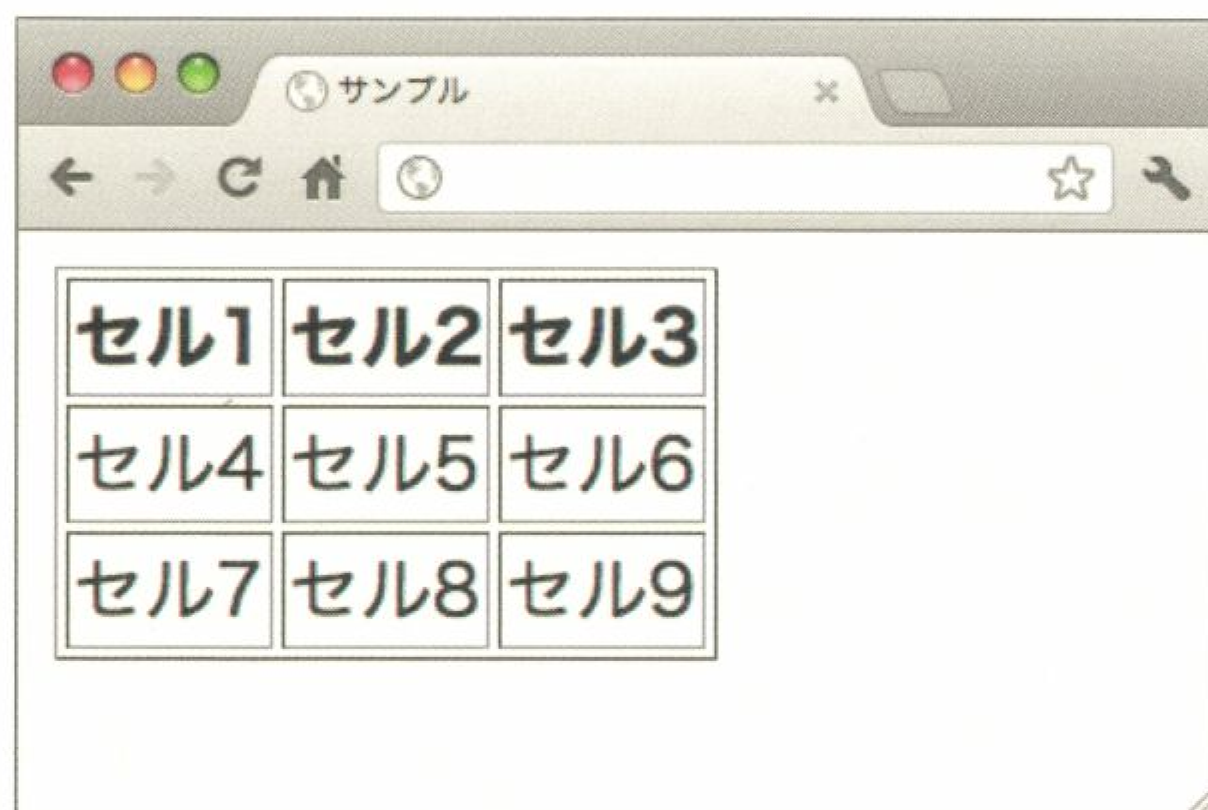
- border="ボーダーの表示"
  この属性はHTML5よりも前の仕様では、テーブル全体を囲う枠線の太さをピクセル数で指定するための属性でした。そのため、多くのブラウザは現在でもこの指定を枠線の太さとして認識します(0より大きい値を指定すると外枠だけでなくセルを区切る枠も表示されます)。しかしHTML5では、この属性はテーブルをレイアウトのために使用していないことを明示的に示す目的で使用されます。その場合、値には1を入れるか、値を空にした状態で指定してください。

次のサンプルは、テーブルの基本構造を具体的にマークアップした例です。

HTML　sample/chapter-09/lecture-9-3/01.html

```
<table border="1">
<tr><th>セル1</th><th>セル2</th><th>セル3</th></tr>
<tr><td>セル4</td><td>セル5</td><td>セル6</td></tr>
<tr><td>セル7</td><td>セル8</td><td>セル9</td></tr>
</table>
```

テーブルの基本構造のマークアップ例



上のソースコードを表示させたところ

▶▶ th要素とtd要素

実はtable要素だけでなく、th要素とtd要素にも専用の属性が用意されています。指定できる属性はほぼ共通しているのですが、ここではセルを連結させるcolspan属性とrowspan属性に注目してください(このあとに具体例のサンプルがあります)。

th要素に指定できる属性

・**colspan="連結させるセルの数"**
この属性を指定したセルから右方向にいくつ分のセルを連結するのかを、1以上の整数で指定します。

・**rowspan="連結させるセルの数"**
この属性を指定したセルから下方向にいくつ分のセルを連結するのかを、1以上の整数で指定します。

・**headers="このセルの見出しセルのid"**
このセルの見出しとなっているセルが音声ブラウザなどでも明確に分かるようにする目的で、見出しとなっているセルに指定されているid属性の値を指定します。id属性の値は半角スペースで区切って複数指定することができます。

・**scope="この見出しセルの対象"**
この見出しセルの対象となっているセルの範囲を示すキーワードを指定します。指定できるキーワードは以下の通りです。

キーワード	見出しの対象となるセルの範囲
row	この見出しセルの右にあるセル全部
col	この見出しセルの下にあるセル全部
rowgroup	この見出しセルの右以降にある同じ横列グループ(thead要素・tbody要素・tfoot要素)のセル
colgroup	この見出しセルの下以降にある同じ縦列グループ(colgroup要素)のセル

td要素に指定できる属性

・**colspan="連結させるセルの数"**
この属性を指定したセルから右方向にいくつ分のセルを連結するのかを、1以上の整数で指定します。

・**rowspan="連結させるセルの数"**
この属性を指定したセルから下方向にいくつ分のセルを連結するのかを、1以上の整数で指定します。

・**headers="このセルの見出しセルのid"**
このセルの見出しとなっているセルが音声ブラウザなどでも明確に分かるようにする目的で、見出しとなっているセルに指定されているid属性の値を指定します。id属性の値は半角スペースで区切って複数指定することができます。

▶▶ セルを連結させる

colspan属性は、そのセルから右側に向かって、指定した数のセルを連結します。同様に、rowspan属性は、そのセルから下側に向かって、指定した数のセルを連結します。これに関しては文章で説明しても分かりにくいので、具体的なサンプルのソースコードで確認してみましょう。

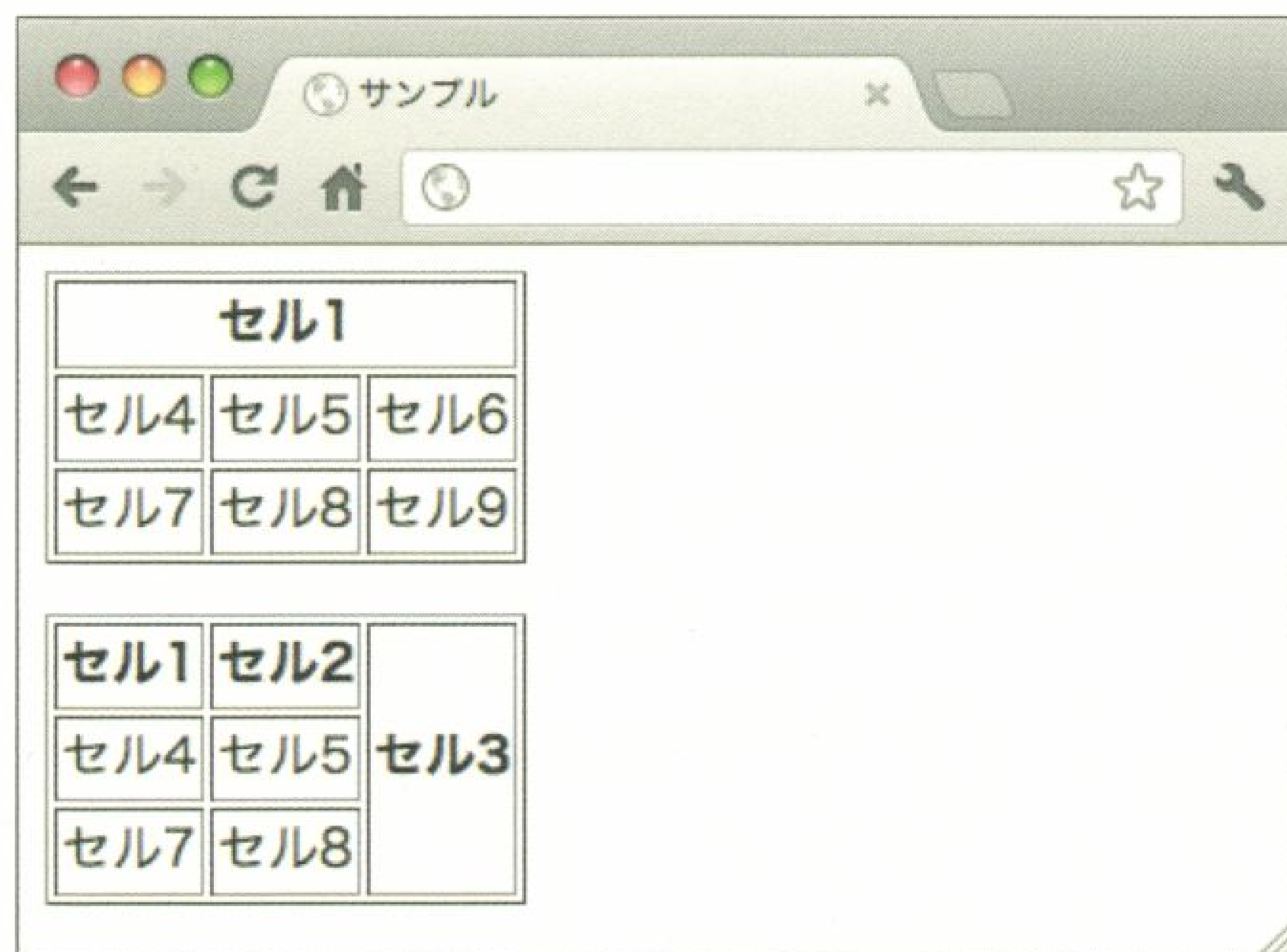
実際には「連結する」というよりも、colspan属性またはrowspan属性を指定したセルが、指定された分だけ拡張されて大きくなると言った方が正しいかもしれません。なぜなら、セルが拡張されることによって表示される場所がなくなったセルのタグは、ソースコードから取り除く必要があるからです。たとえば以下のHTMLのソースコードにおいて、上のテーブルではセル2とセル3の要素が取り除かれています。同様に、下のテーブルではセル6とセル9の要素が取り除かれています。

HTML　　**sample/chapter-09/lecture-9-3/02.html**

```
<table border="1">
<tr><th colspan="3">セル1</th></tr>
<tr><td>セル4</td><td>セル5</td><td>セル6</td></tr>
<tr><td>セル7</td><td>セル8</td><td>セル9</td></tr>
</table>

<table border="1">
<tr><th>セル1</th><th>セル2</th><th rowspan="3">セル3</th></tr>
<tr><td>セル4</td><td>セル5</td></tr>
<tr><td>セル7</td><td>セル8</td></tr>
</table>
```

colspan属性とrowspan属性を使用したマークアップ例



上のサンプルの表示結果

▶▶ テーブルにキャプションをつける

テーブルには、キャプション（タイトル）をつけることができます。キャプションは**caption要素**としてマークアップし、<table>〜</table>の範囲内の先頭に入れます。

▶▶ セルを横にグループ化する

また、tr要素はテーブル内のヘッダー／ボディ／フッターのいずれかとしてグループ化することができます。その際に使用するのが、**thead要素**（table header）・**tbody要素**（table body）・**tfoot要素**（table footer）です。これらの要素を使用する場合、その順序は基本的にはthead要素・tfoot要素・tbody要素のように、tbody要素よりもtfoot要素を先にする必要がある点に注意してください。これは、かなり長いかもしれないtbody要素よりも先にtfoot要素を読み込んで表示できるようにする目的で、HTML4で決められた仕様です。ただし、HTML5の仕様ではthead要素・tbody要素・tfoot要素の順に配置することも認められています。

HTML 　sample/chapter-09/lecture-9-2/03.html

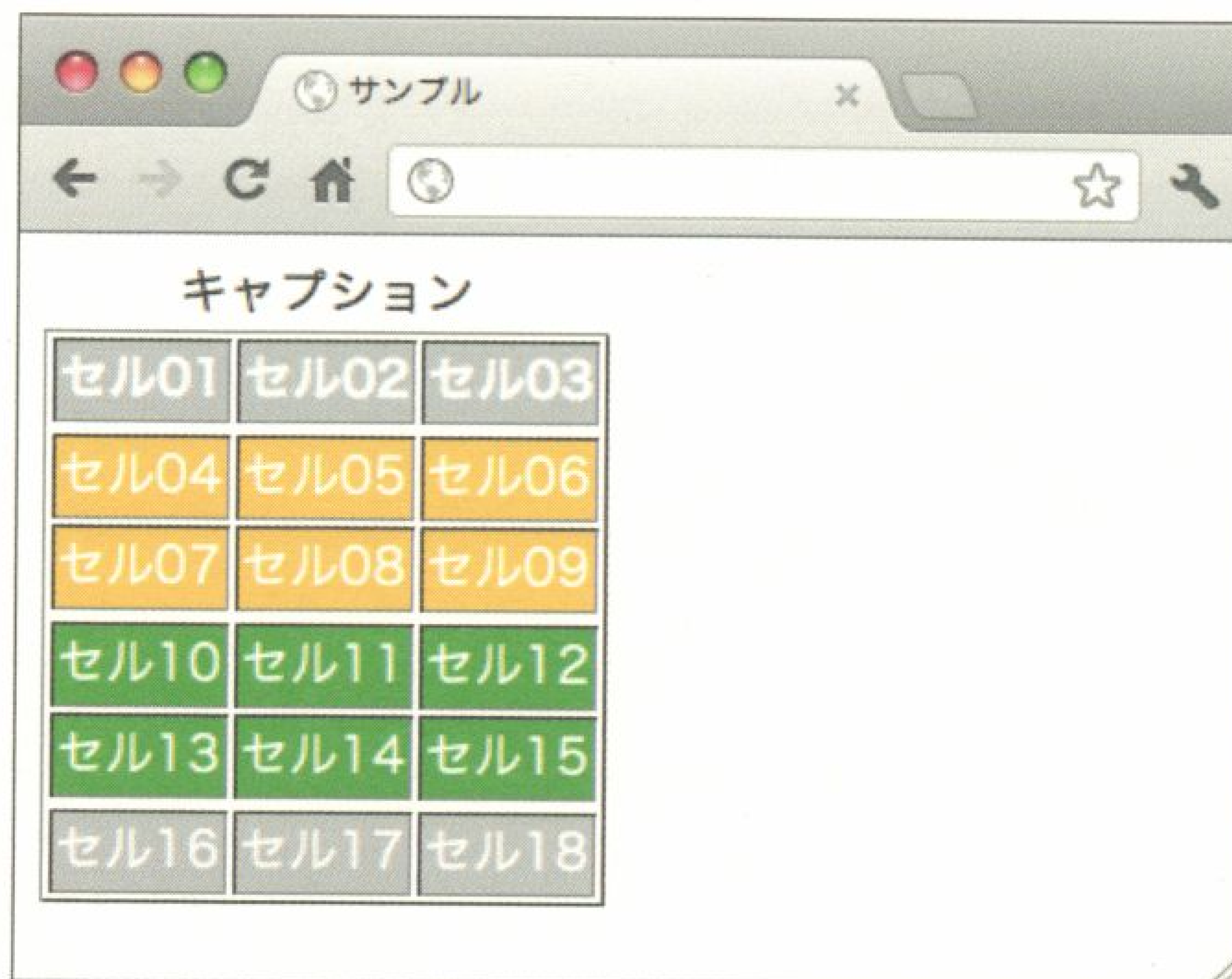
```html
<table border="1">
<caption>キャプション</caption>
<thead>
<tr><th>セル01</th><th>セル02</th><th>セル03</th></tr>
</thead>
<tfoot>
<tr><td>セル16</td><td>セル17</td><td>セル18</td></tr>
</tfoot>
<tbody id="tb1">
<tr><td>セル04</td><td>セル05</td><td>セル06</td></tr>
<tr><td>セル07</td><td>セル08</td><td>セル09</td></tr>
</tbody>
<tbody id="tb2">
<tr><td>セル10</td><td>セル11</td><td>セル12</td></tr>
<tr><td>セル13</td><td>セル14</td><td>セル15</td></tr>
</tbody>
</table>
```

CSS 　sample

```css
thead th, tfoot td {
  color: #fff;
  background: #bbb;  /* グレー */
}
tbody#tb1 td {
  color: #fff;
  background: #fc0;  /* 黄色 */
}
```

```
tbody#tb2 td {
  color: #fff;
  background: #390;  /* 緑 */
}
```

caption要素およびthead要素・tbody要素・tfoot要素を使用した例



上のサンプルの表示結果

LECTURE 9-4

テーブル関連のプロパティ

続いてテーブルに関連するCSSプロパティについて説明していきます。

隣接するボーダーを1本の線にする

テーブルの初期状態では、table要素のボーダーとth要素またはtd要素のボーダーがそれぞれ独立したボーダーとして個別に表示されるようになっています。**border-collapseプロパティ**を使用すると、それぞれの隣接するボーダーをまとめて1本の線にして表示させることができます。

border-collapseに指定できる値

- `collapse`
  テーブル内の隣接するボーダーはすべてまとめて1本にして表示させます。

- `separate`
  テーブル全体のボーダーと各セルのボーダーをそれぞれ独立したものとして別々に表示させます。

HTML　sample/chapter-09/lecture-9-4/01.html

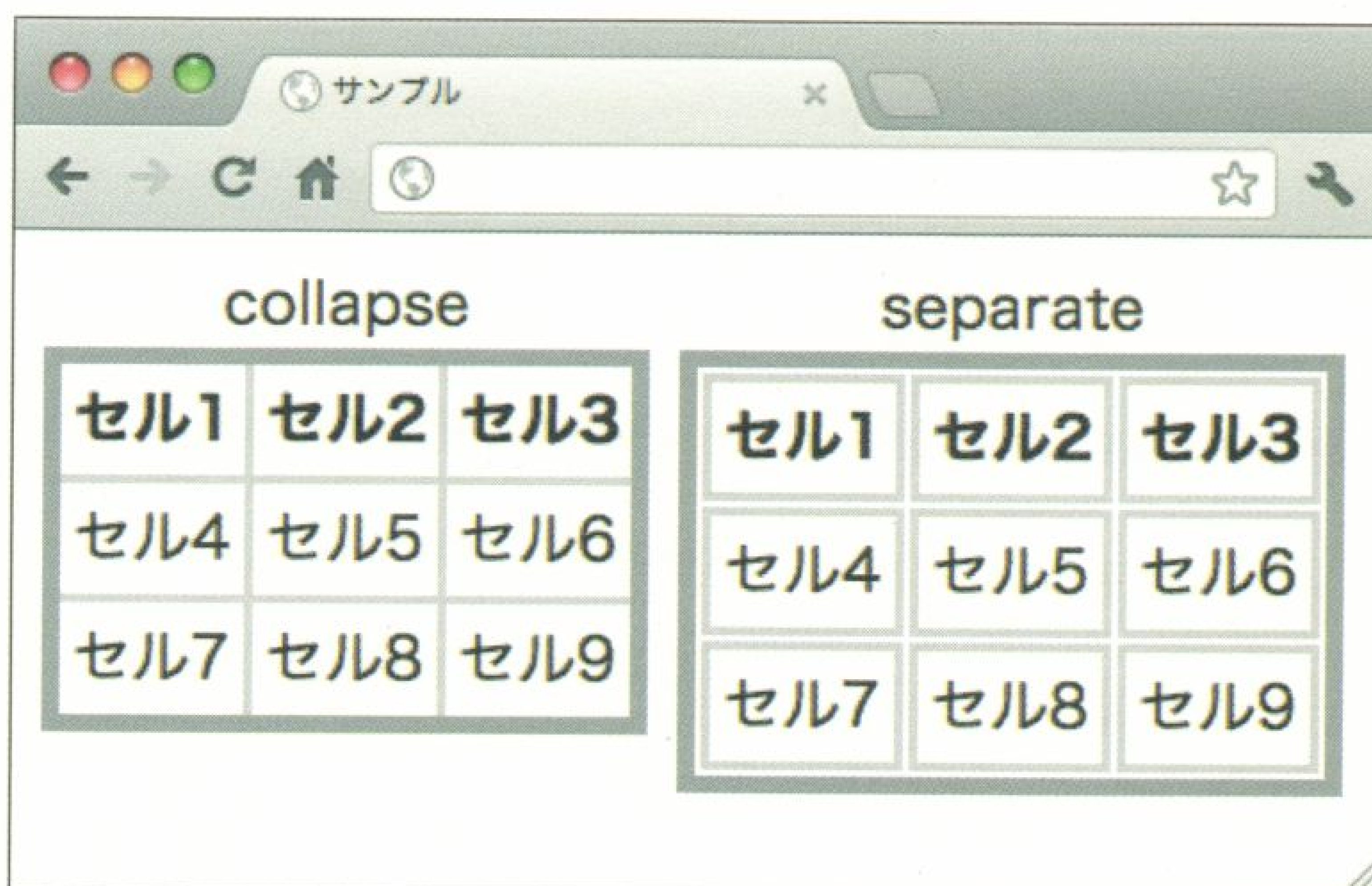
```
<table border="1" id="sample1">
<caption>collapse</caption>
<tr><th>セル1</th><th>セル2</th><th>セル3</th></tr>
<tr><td>セル4</td><td>セル5</td><td>セル6</td></tr>
<tr><td>セル7</td><td>セル8</td><td>セル9</td></tr>
</table>

<table border="1" id="sample2">
<caption>separate</caption>
<tr><th>セル1</th><th>セル2</th><th>セル3</th></tr>
<tr><td>セル4</td><td>セル5</td><td>セル6</td></tr>
<tr><td>セル7</td><td>セル8</td><td>セル9</td></tr>
</table>
```

CSS　sample/chapter-09/lecture-9-4/01-border-collapse.css

```
table {
  float: left;
  border: 5px solid #999;
}
th, td {
  padding: 0.2em;
  border: 3px solid #ccc;
}
table#sample1 {
  border-collapse: collapse;
}
table#sample2 {
  border-collapse: separate;
  margin-left: 8px;
}
```

border-collapseプロパティの使用例



上のソースコードの表示結果

テーブルのキャプション位置を下にする

caption-sideプロパティを使用すると、通常はテーブルの上に表示されるキャプションを、テーブルの下に表示させることができます。

caption-sideに指定できる値

- **top**
  キャプションをテーブルの上に表示させます。

caption-sideに指定できる値(続き)

・bottom
　キャプションをテーブルの下に表示させます。

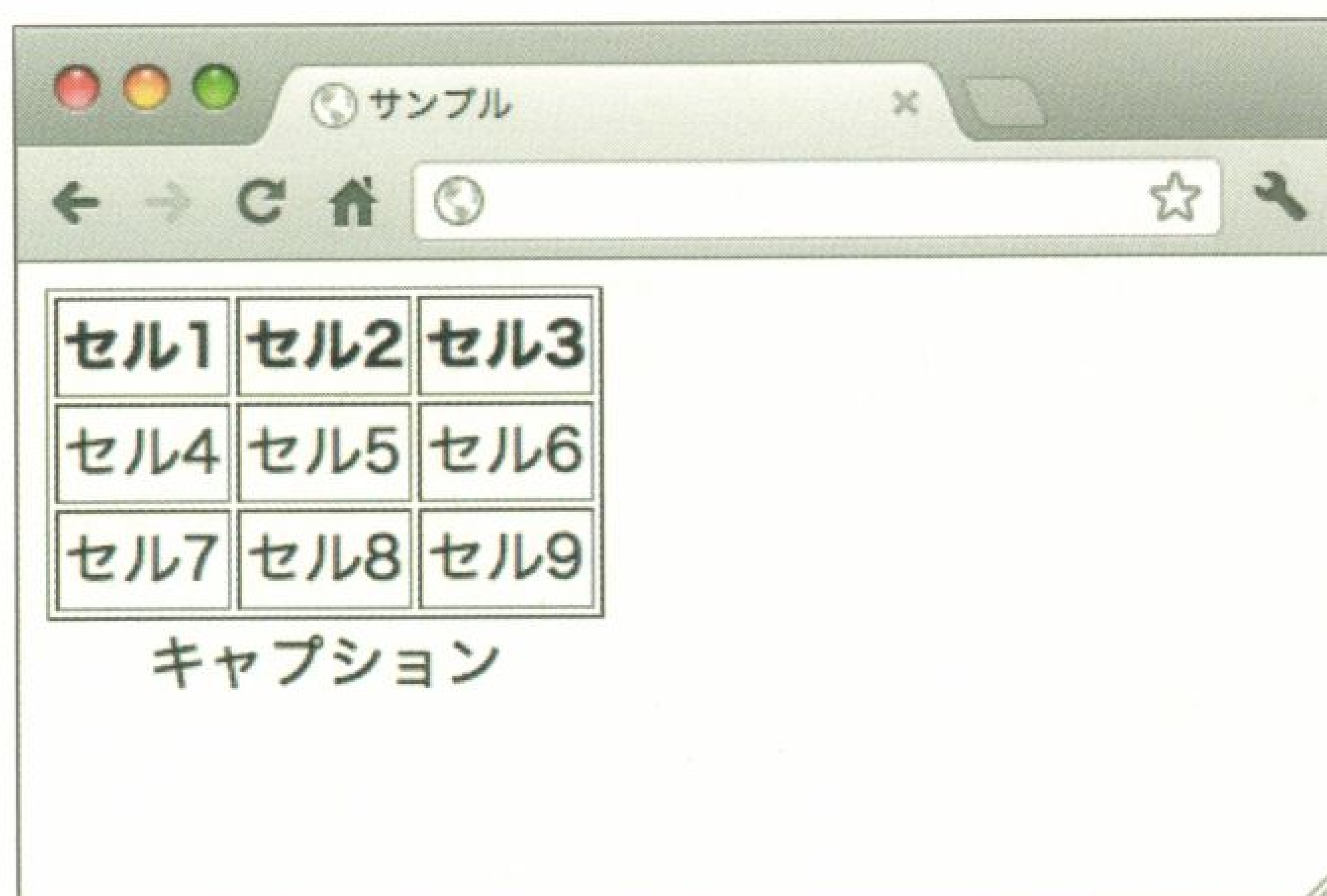
HTML　sample/chapter-09/lecture-9-4/02.html

```
<table border="1">
<caption>キャプション</caption>
<tr><th>セル1</th><th>セル2</th><th>セル3</th></tr>
<tr><td>セル4</td><td>セル5</td><td>セル6</td></tr>
<tr><td>セル7</td><td>セル8</td><td>セル9</td></tr>
</table>
```

CSS　sample/chapter-09/lecture-9-4/02-caption-side.css

```
caption {
  caption-side: bottom;
}
```

caption-sideプロパティの使用例



上のソースコードの表示結果

CHAPTER 10

その他の機能とテクニック

ここまでで、現時点で利用可能な主要機能はほぼ解説しましたが、覚えておくべきことはもう少しあります。Chapter 10では、ここまでに登場しなかった要素とプロパティ、フロートをクリアするための特殊なテクニック、出力先のメディア特性に合わせてCSSを切り替える方法について解説します。

LECTURE 10-1

その他の要素

ここでは、Chapter 9までには登場しなかったけれど、覚えておくべき要素について説明していきます。それぞれ役目が異なりますが、使用頻度は高いので、頭に入れておきましょう。

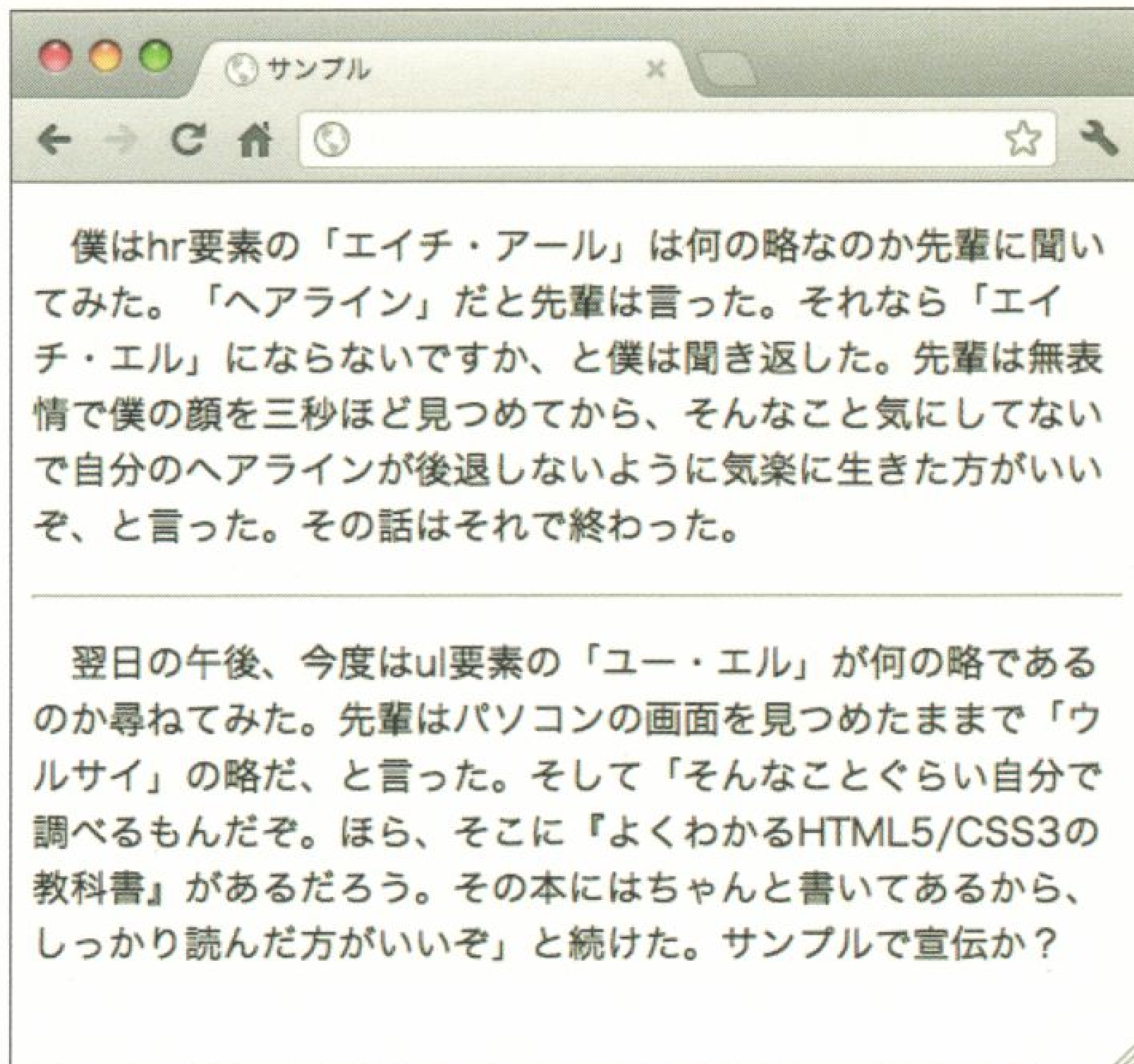
区切り線 HTML5改

hr要素の「hr」は「horizontal rule（横罫線）」の略で、もともとは単純に**横線を表示させる**だけの空要素でした。しかしHTML5からはその役割が変更され、線を表示させることが目的なのではなく、その部分で**話題（または物語の場面など）が変わっている**ことを示すことが目的の要素となりました。とはいっても、セクションレベルでの大きな主題の変わり目で使用するのではなく、**段落レベルでの小さな変わり目**に使用することが想定されています。このように役割は変更されましたが、現在でも一般的なブラウザにおいては横線として表示されることに変わりはありません。

HTML　　sample/chapter-10/lecture-10-1/01.html

```
<p>
　僕はhr要素の「エイチ・アール」は何の略なのか先輩に聞いてみた。「ヘアライン」だと先輩は言った。それなら「エイチ・エル」にならないですか、と僕は聞き返した。先輩は無表情で僕の顔を三秒ほど見つめてから、そんなこと気にしてないで自分のヘアラインが後退しないように気楽に生きた方がいいぞ、と言った。その話はそれで終わった。
</p>
<hr>
<p>
　翌日の午後、今度はul要素の「ユー・エル」が何の略であるのか尋ねてみた。先輩はパソコンの画面を見つめたままで「ウルサイ」の略だ、と言った。そして「そんなことぐらい自分で調べるもんだぞ。ほら、そこに『よくわかるHTML5/CSS3の教科書』があるだろう。その本にはちゃんと書いてあるから、しっかり読んだ方がいいぞ」と続けた。サンプルで宣伝か？
</p>
```

hr要素の使用例。この例では、場面が変わる部分で使用している



前ページのソースコードを表示させたところ

追加と削除

ins要素はあとから**追加**された部分（inserted text）、**del要素**はあとから**削除**された部分（deleted text）をあらわす要素です。一般的なブラウザでは、ins要素は**下線付きの状態**で、del要素は**取消線が引かれた状態**で表示されますが、表示方法についてはCSSで自由に変更できます。これらの要素は、インラインの範囲にでもブロックレベルの範囲にでもマークアップできます。次の属性が指定できます。

ins要素・del要素に指定できる属性

- **cite="ファイルのURL"**
  追加・削除した理由等が書かれているページのURLを指定します。

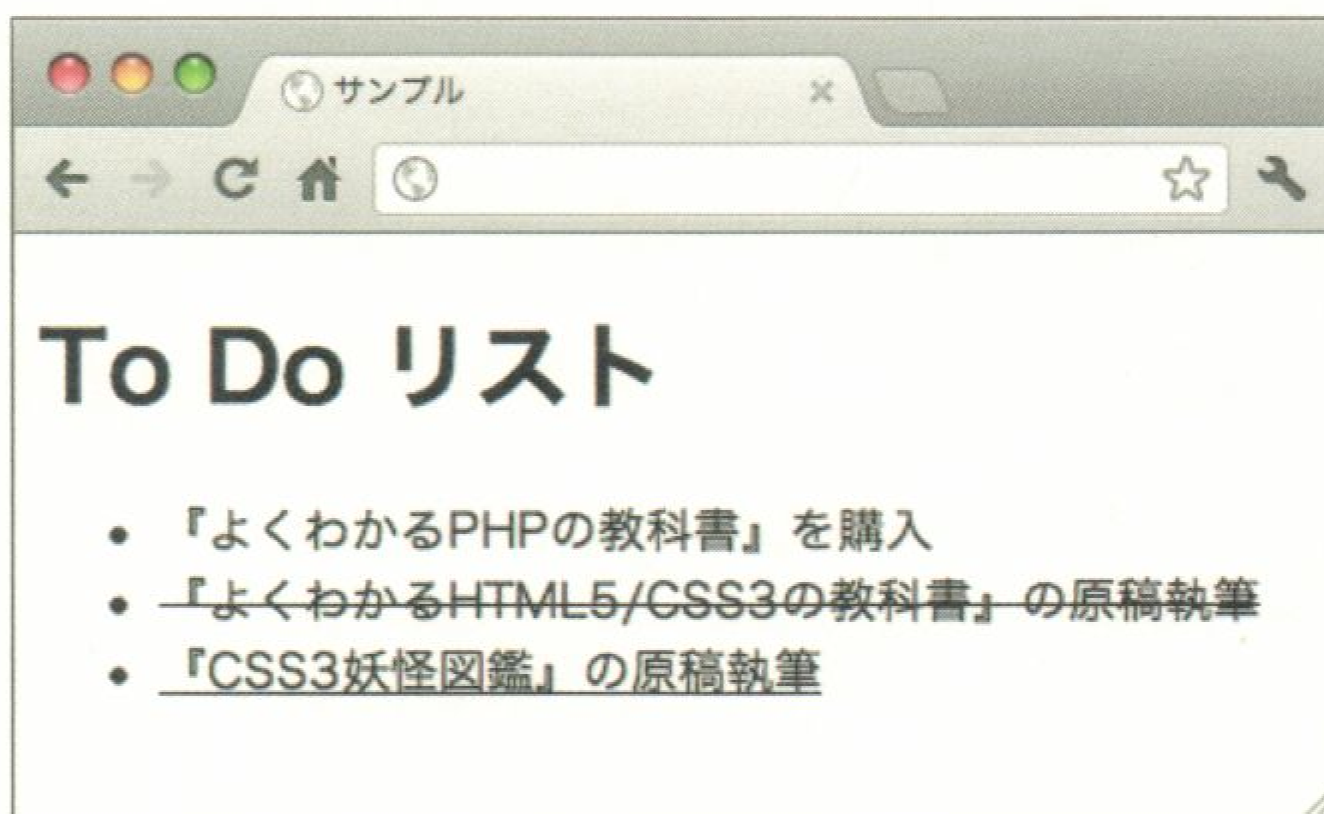
- **datetime="追記・訂正した日時"**
  追加・削除した日時を「YYYY-MM-DDThh:mm:ssTZD」の書式で指定します。書式は「年4桁-月2桁-日2桁T時2桁:分2桁:秒2桁タイムゾーン」の意味です。大文字の「T」はそのまま固定で、タイムゾーンは日本の場合「+09:00」となります。たとえば、日本時間の「2012年10月13日15時20分30秒」は「2012-10-13T15:20:30+09:00」となります。

HTML　sample/chapter-10/lecture-10-1/02.html

```
<h1>To Do リスト</h1>

<ul>
<li>『よくわかるPHPの教科書』を購入</li>
<li><del datetime="2012-06-18T02:50:00+09:00">『よくわかるHTML5/CSS3の教科書』の原稿
執筆</del></li>
<li><ins datetime="2012-07-16T10:30:00+09:00">『CSS3妖怪図鑑』の原稿執筆</ins></li>
</ul>
```

ins要素とdel要素の使用例



上のソースコードを表示させたところ

スクリプト

script要素は、HTMLにスクリプトを組み込むための要素です。CSSの場合は、要素内容としてCSSを書き込むなら**style要素**、CSSのファイルを読み込むなら**link要素**と使い分けが必要でしたが、script要素の場合はこれだけで要素内容として**スクリプトを書き込む**ことも**外部のファイルを読み込む**ことも可能です。次の属性が指定できます。

ただし、要素内容としてスクリプトを書き込むことができるのは、**src属性**を指定していない場合に限られます。src属性を指定した場合は、要素内容は空にしておかなければなりません（コメントによる注意書きなどは入れられます）。

script要素に指定できる属性

・src="ファイルのURL"
　スクリプトを記述したファイルのURLを指定します。

script要素に指定できる属性(続き)

- **type="MIMEタイプ"**
  スクリプト言語のMIMEタイプを指定できます。script要素はもともとJavaScript専用ではなく、JavaScript以外のスクリプト言語にも対応できるようになっているため、この属性が用意されています。この属性を指定しなかった場合は「text/javascript」が指定された状態となります。

- **charset="文字コード"**
  src属性で指定しているファイルの文字コードを示します。

HTML

```
<script src="js/example.js"></script>

<script>
  ～スクリプト～
</script>
```

スクリプトは、このいずれかのパターンで記述する

▶▶ スクリプトが動作しない環境向けには

スクリプトはすべてのユーザーの環境で動作するわけではありません。意図的にスクリプトが動作しないように設定している人もいますし、そもそもスクリプトが動作しない(一般的ではない)ブラウザを使用している人もいます。そのようなスクリプトが動作しない環境向けの内容を別途用意しておきたい場合には、**noscript要素**を使用します。この要素の要素内容は、スクリプトが動作する環境では無視されますが、動作しない環境においては有効となります。

noscript要素は、body要素内で使用できるだけでなく、head要素内で使用することも可能です。その場合は内容としてlink要素・style要素・meta要素が入れられます。

CHAPTER 10

インラインフレーム

iframe要素を使用すると、Webページの中に別のWebページをインラインの状態で表示させることができます(iframeはinline frameの略です)。要素内容は、インラインフレームが表示できないときに限り表示されます。この要素は、YouTubeの動画や一部の広告などをWebページに組み込む際に利用されています。

iframe要素に指定できる属性

・**src="ページのURL"**
インラインフレームの中に表示させるページのURLを指定します。

・**width="幅"**
インラインフレームの幅をピクセル数(単位をつけない整数)で指定します。

・**height="高さ"**
インラインフレームの高さをピクセル数(単位をつけない整数)で指定します。

・**name="フレーム名"**
インラインフレームの名前を指定します。この名前は、リンク先を表示させるフレームとしてa要素のtarget属性の値などで指定できます。

HTML(03.html)　　sample/chapter-10/lecture-10-1/03.html

```
・・・
<body>

<iframe src="03-2.html" width="250" height="150">
</iframe>

</body>
</html>
```

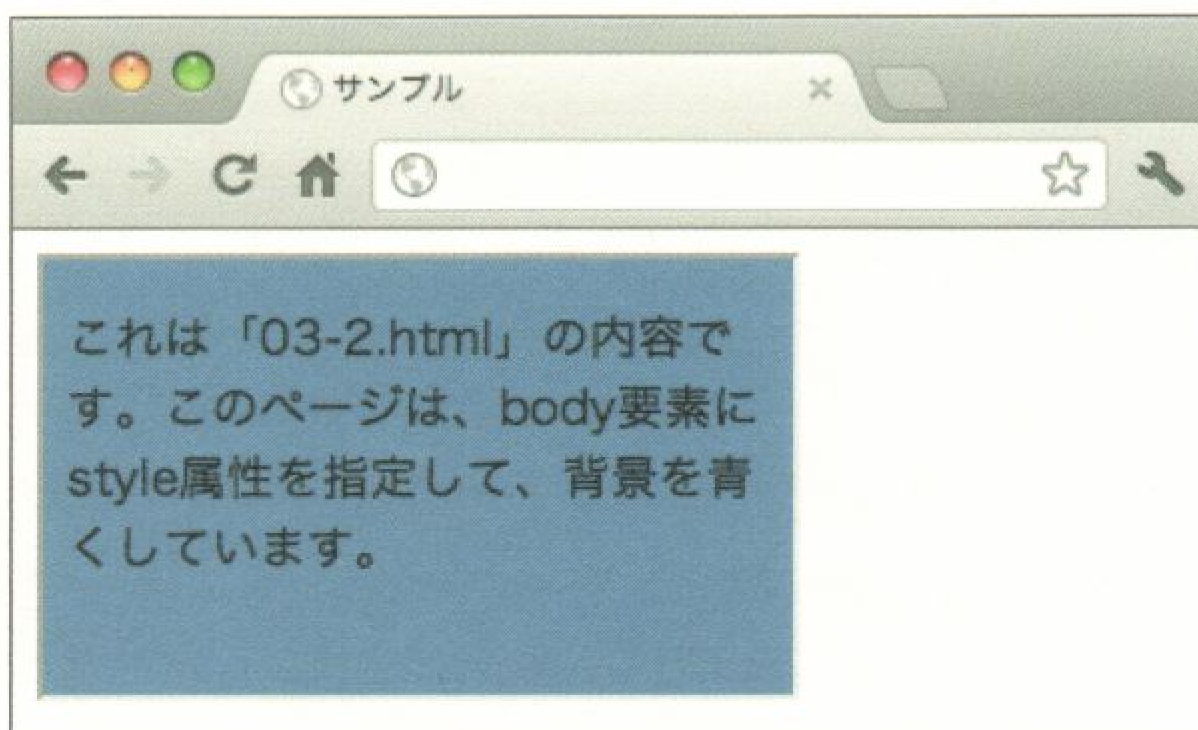
HTML(03-2.html)　　sample/chapter-10/lecture-10-1/03-2.html

```
・・・
<body style="background: steelblue">

<p>
これは「03-2.html」の内容です。このページは、body要素にstyle属性を指定して、背景を
青くしています。
</p>

</body>
</html>
```

iframe要素の使用例



上の「03.html」を表示させたところ。インラインフレームの内容として「03-2.html」が表示されている

LECTURE 10-2

その他のプロパティ

続いて、Chapter 9までに登場しなかったプロパティについて説明していきます。コンテンツを追加するcontentプロパティと、引用符を追加するquotesプロパティについてです。

コンテンツの追加

contentプロパティを使用すると、HTML内には存在していないコンテンツをCSSで追加することができます。コンテンツは、セレクタの**::before疑似要素**または**::after疑似要素**を使用して、指定した要素内の内容全体の直前または直後に挿入されます。追加するコンテンツは、**displayプロパティ**を使用してインラインにでもブロックレベルにでもできます。次の値が指定できます。

contentに指定できる値

- **テキスト**
  コンテンツとして追加するテキストを二重引用符(")または一重引用符(')で囲って指定します。

- **url(データのアドレス)**
  コンテンツとして追加するデータ(画像など)のアドレスを指定します。

- **attr(属性名)**
  このプロパティが指定された要素に、属性名で指定した属性が指定されている場合、その値をテキストとして追加します。

- **open-quote, close-quote**
  次に解説するquotesプロパティで設定されている引用符を追加します。

- **none**
  コンテンツを追加しません。

HTML　sample/chapter-10/lecture-10-2/01.html

```
<h1>CSSで内容を追加する方法</h1>
```

CSS　sample/chapter-10/lecture-10-2/01-content.css

```
h1 {
  font-size: large;
}
h1::before {
  content: "特別コラム：";
  display: block;
  font-size: small;
  color: #f6d;
}
h1::after {
  content: url(sakura.jpg);
  display: block;
}
```

contentプロパティの使用例



上のソースコードを表示させたところ

引用符の設定

quotesプロパティは、contentプロパティで追加する**引用符（open-quote、close-quote）を指定する**プロパティです。引用部分の前につける記号と後につける記号を半角スペースで区切ってペアで指定します。さらに半角スペースで区切って記号のペアを指定することで、引用が入れ子になった場合に使用する記号を**いくつでも**指定できます（入れ子の深さに応じて、次々に右側のペアの記号が採用されます）。

quotesに指定できる値

- **文字列**
  引用符として使用する記号を半角スペースで区切ってペアで指定します。さらに半角スペースで区切ってペアを指定しておくと、引用が入れ子になった場合の引用符として使用されます。

- **none**
  引用符を表示しません。

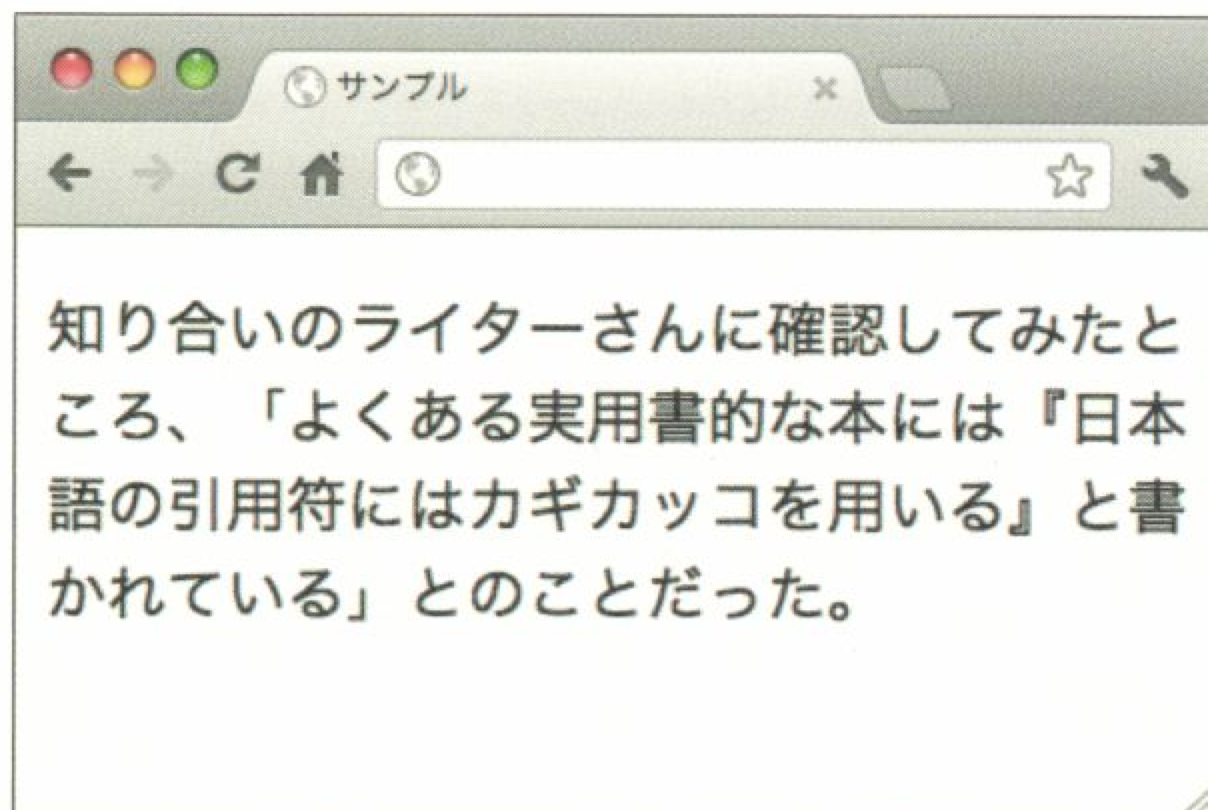
HTML　sample/chapter-10/lecture-10-2/02.html

```
<p>
知り合いのライターさんに確認してみたところ、<q>よくある実用書的な本には<q>日本語の引用符にはカギカッコを用いる</q>と書かれている</q>とのことだった。
</p>
```

CSS　sample/chapter-10/lecture-10-2/02-quotes.css

```
q:before { content: open-quote; }
q:after  { content: close-quote; }
q { quotes: '「' '」' "『" "』"; }
```

quotesプロパティの使用例



上のソースコードを表示させたところ

LECTURE 10-3

clearfixについて

続いてclearfixについて説明します。clearfixは、フロートの不都合を解消するために生まれたテクニックです。時代とともにその形は変化してきて、いくつかの対処方法があります。どのような問題に対して、どういう方法で対処しているのか、少しややこしいので、ここでしっかりと理解しておきましょう。

フロートで不都合なこと

ここでは、フロートをクリアするための特殊なCSSのテクニック「clearfix」について説明します。floatプロパティやclearプロパティは、もともと画像などの横にテキストを回り込ませる目的で用意されたものであるため、それを段組みのようにボックスを横に並べる目的で使用していると不都合が生じることがあります。次の例を見てください。これはChapter 7で使った2段組みのサンプルをさらに単純にしたものです。

HTML　sample/chapter-10/lecture-10-3/01.html

```
<div id="contents">

<div id="main">
メインの段のテキストです。
メインの段のテキストです。
メインの段のテキストです。
メインの段のテキストです。
メインの段のテキストです。
</div>

<div id="sub">
サブの段のテキストです。
サブの段のテキストです。
サブの段のテキストです。
</div>

</div>
```

CSS　sample/chapter-10/lecture-10-3/01-2cols.css

```
#contents {
  margin: 0 auto;
  width: 300px;
}

#main {
  float: right;
  width: 200px;
  color: #fff;
  background: #fc0;       /* 黄色 */
}

#sub {
  float: left;
  width: 100px;
  color: #fff;
  background: #390;       /* 緑 */
}
```

シンプルな2段組みのソース例



上のソースコードの表示

では、#contentsに赤いボーダーを表示させて、全体を囲ってみましょう。CSSの#contentsに対する指定のところに、次のようにボーダーを表示させる指定を追加してください。

CSS

```
#contents {
  margin: 0 auto;
  width: 300px;
  border: 10px solid red;   ←―― 追加
}
```

全体を囲う#contentsに赤いボーダーを表示させる

ソースコードを保存してブラウザで最表示させると、ボーダーはこんな風に表示されてしまいました。なぜ、赤いボーダーは全体を囲うように表示されないのでしょうか？



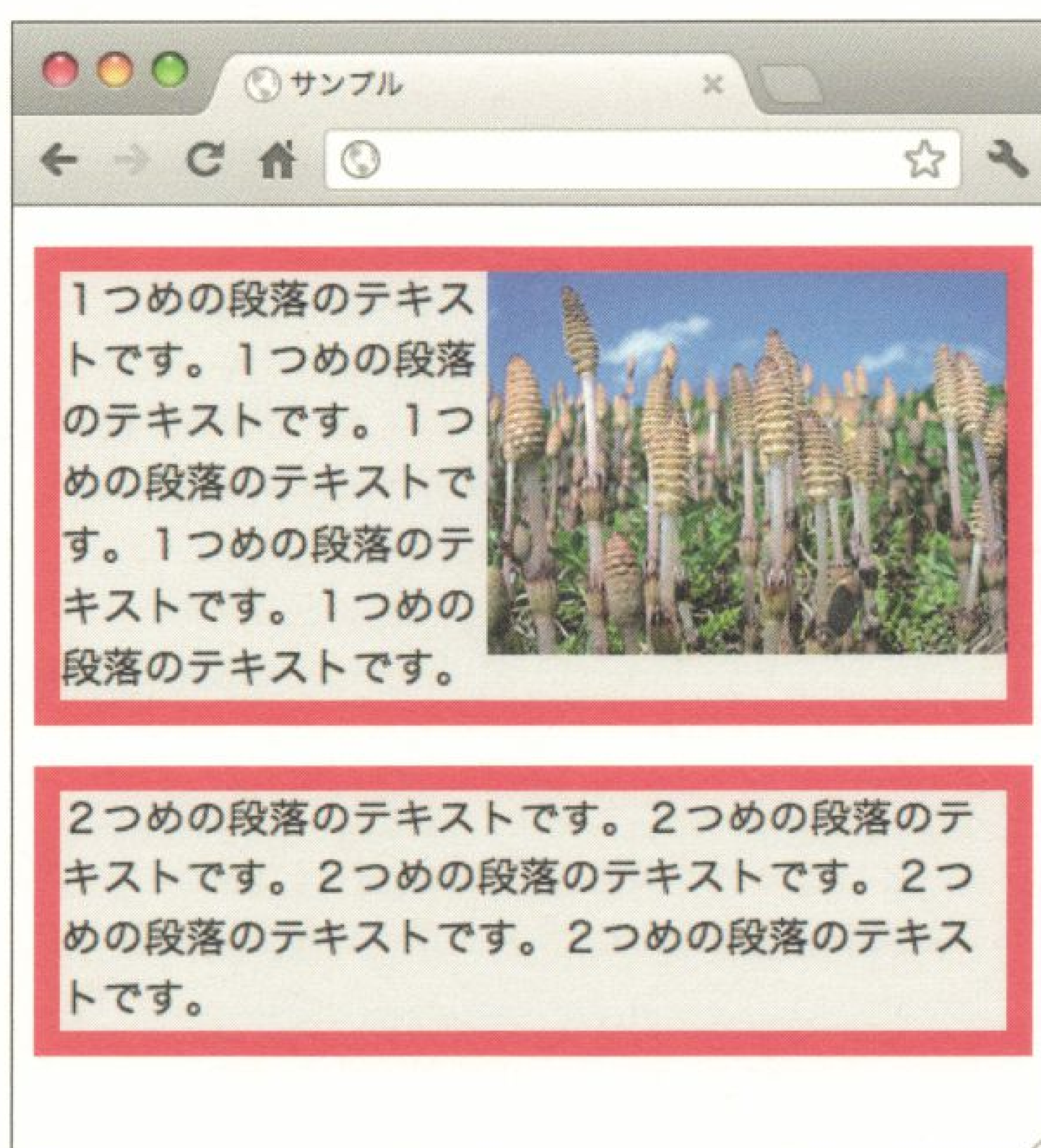
#contentsに赤いボーダーを表示させたところ

▶▶ floatを指定した要素は親要素からはみ出す

実は、floatを指定した要素のボックスは、それを含む要素のボックスの高さとは無関係になって、そこからはみ出す仕様となっています。つまり、上のサンプルでは、#contentsの内容は2つともfloatが指定されているので#contentsからはみ出し、#contentsの**高さは0**になっているのです。

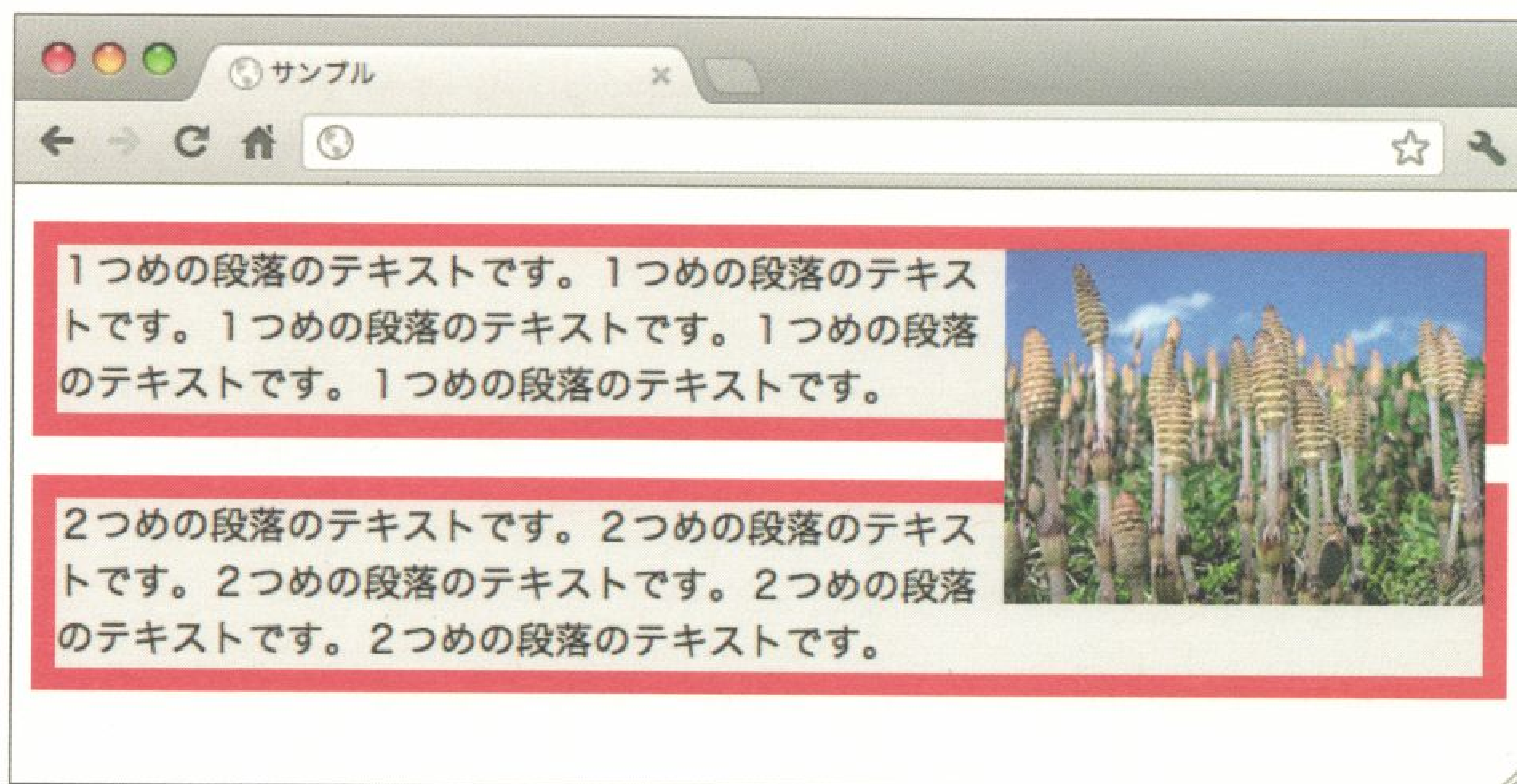
これは、画像の横にテキストを回り込ませるという本来の機能を考えると理解できます。

たとえば、下のスクリーンショットは、2つの段落のうち1つめの段落の先頭に画像を入れて、それに「float: right;」を指定したものです。ボックスの状態が分かりやすいように赤いボーダーとグレーの背景を指定しています。



赤いボーダーを指定した2つの段落があり、先頭の画像は右にフロートしている

では、このウィンドウを横に広げて、2つめの段落が画像の横に回り込む様子を見てみましょう。画像は1つめの段落のボックスをはみ出し、2つめの段落に入り込んでいます。ボックスの形状は常に四角形であるため、画像の横に後続のボックスのテキストを回り込ませるには、フロートした画像はこのようにボックスからはみ出る必要があるのです。



ウィンドウの幅を広げると画像は1つめの段落をはみ出し、2つめの段落に入り込む

▶▶ clear要素を使用した場合の不都合

単純にコンテンツを2段組みにするだけであれば、このような仕様でも特に問題はないかもしれません。しかし、段組み部分を囲う親ボックスにボーダーを表示させたり背景を表示させたりして、その中にフロートさせた要素がすっぽりと収まるようにしたい場合には、ちょっと手間がかかります。

なにしろclear要素はブロックレベル要素の先頭でフロートを解除するプロパティです。2つめの段落の先頭でフロートをクリアしたところで、2つめの段落は回り込まなくはなりますが、1つめの段落からフロートがはみ出す状態に変化はありません。1つめの段落からフロートがはみ出ないようにするには、1つめの段落内の最後でフロートをクリアする必要があります。つまり、1つめの段落の最後に、フロートをクリアするためだけにブロックレベル要素を追加する必要があるのです(clearプロパティの適用対象はブロックレベル要素のみであるため)。ところが、p要素の中にはブロックレベル要素は入れられない仕様になっています。こんなときはどうすればいいのでしょうか？

CHAPTER 10

フロートの不都合を解消する（1）

実はこの問題を解消するシンプルで簡単な方法があります。それは、フロートを含む**親要素自体**もフロートさせてしまうか、overflowプロパティで「visible」**以外の値を指定する**ことです。こうするだけで、フロートさせた要素の親要素は、フロートしたボックス全体を含むように拡張されるのです。

しかし、親要素もフロートさせるということは、場合によってはさらにそれをクリアする必要が出てきます。つまり、問題を一階層外側に追いやっただけの状態になる可能性があるということです。ですので、一般的にはフロートを使う方法はあまり利用されていません。

というわけで、最近ではoverflowプロパティに「hidden」または「auto」を指定して問題を解消するケースが多くなっています。「最近多くなっている」という言い方をしたのは、かつてはこの問題を解消するためにoverflowプロパティはあまり使用されていなかったからです。それは、overflowプロパティを使った方法では、Nescape 6やOpera6～7といった一部の古いブラウザには対応できなかったという理由によります。現在では、これらのブラウザのことはほとんど気にする必要もありませんので、**「overflow: hidden;」を指定すれば問題は解消できます。**

CSS　sample/chapter-07/lecture-7-5/02-2cols.css（HTMLファイルは02.html）

```
#contents {
  margin: 0 auto;
  width: 300px;
  border: 10px solid red;
  overflow: hidden;   ← 追加
}
```

先程のサンプルに「overflow: hidden;」を追加した例



赤いボーダーが、フロートしたボックス全体を囲うようになっている

フロートの不都合を解消する（2）

では当時、Nescape 6やOpera6〜7といった一部の古いブラウザでも問題を解消させるために、どのような方法がとられていたのでしょうか？　実は、contentプロパティとdisplayプロパティを使って、フロートを含む親要素の最後の部分にブロックレベル要素を生成させ、そこにclearプロパティを指定して解除させるという手法がとられていたのです。しかも、単純にそれをおこなうだけではうまく対応できないブラウザもあったため、様々な部分でチューニングをおこない、どのブラウザでもうまく動くようになっていました。そのCSSの裏技的ソースコードが、clearfixと呼ばれ長いあいだ使われ続けてきたテクニックなのです。

▶▶ clearfixの原型

clearfixは時代とともに変化してきましたが、その原型は次のようなものでした。

CSS

```
.clearfix:after {
  content: ".";
  display: block;
  height: 0;
  clear: both;
  visibility: hidden;
}

.clearfix { display: inline-block; }

/* Hides from IE-mac ¥*/
* html .clearfix { height: 1%; }
.clearfix { display: block; }
/* End hide from IE-mac */
```

clearfixの原型。これだけでIE5（Mac版も含む）やNescape6、Opera6など、当時のほぼすべてのブラウザに対応していた

使用方法は、当時はフロートを含む親要素に「`class="clearfix"`」を直接指定して適用させるか、clearfixのソースコード内の「`.clearfix`」の部分を親要素に既に指定されているクラス名に置き換えて使用していました。

▶▶ 現在のclearfixのコード

clearfixの原型はあまりにも長かったため、時代とともにサポートする必要のなくなったブラウザに対する処理も削られていき、現在ではほぼ次のようなシンプルな形になっているようです(実際にはさまざまなバリエーションが生み出されています)。ただし、Internet Explorer 6にも対応する必要がある場合には、「zoom: 1;」または「width: 100%;」などの指定も別途追加する必要があります。

CSS

```
.clearfix:after {
  content: "";
  display: block;
  clear: both;
}
```

シンプルになった現在のclearfixの例。IE6にも対処する必要がある場合は別途指定が必要

clearfixを使用するには、いちいちクラスをつける必要があったり、一部のHTML編集ソフトのデザインビューで指定通りに表示されないなどの問題もあって、現在ではあまり使われなくなってきています。とはいえ、overflowを使う方法ではうまく対処できないような場合※15には、現在でもclearfixは利用されることがあるようです。

※15：たとえば、内容の一部をフロートとは別にあえてはみ出させているようなデザインの場合には、「overflow: hidden;」や「overflow: auto;」が使えないこともあります。

LECTURE 10-4

メディアクエリー

「メディアクエリー」を使うと、出力する媒体や状態ごとに、適用するCSSを変えることができます。表示領域の幅などに応じてCSSを変えることができる便利な機能です。

メディアクエリーとは？ 【CSS3改】

Chapter 4ではCSSの組み込み方を解説し、その中でlink要素やstyle要素にはmedia属性が指定できるという説明もしました。media属性には次の値が指定でき、それによってCSSの適用対象とする出力媒体を限定したい場合に使用します。

値	意味
all	すべての機器
screen	パソコン画面
print	プリンタ
projection	プロジェクタ
tv	テレビ
handheld	携帯用機器（画面が小さく回線容量も小さい機器）
tty	文字幅が固定の端末（テレタイプやターミナルなど）
speech	スピーチ・シンセサイザー（音声読み上げソフトなど）
braille	点字ディスプレイ
embossed	点字プリンタ

media属性に指定できる値。これによってCSSを適用する出力媒体を限定できる

▶▶CSS3での拡張点

CSS2.1で指定できるのはここまでだったのですが、CSS3からはこの機能が拡張されて、出力媒体の種類だけでなくその媒体の特性や状態を示す式も書き込めるようになっています。これによって、たとえばウィンドウの幅が640ピクセルよりも小さければこのCSSを適用し、640ピクセル以上であればこのCSSを適用する、といった指定が可能になります。このように、出力媒体の特性や状態を式にあらわして適用するCSSを指定できる機能のことをメディアクエリーと言います。

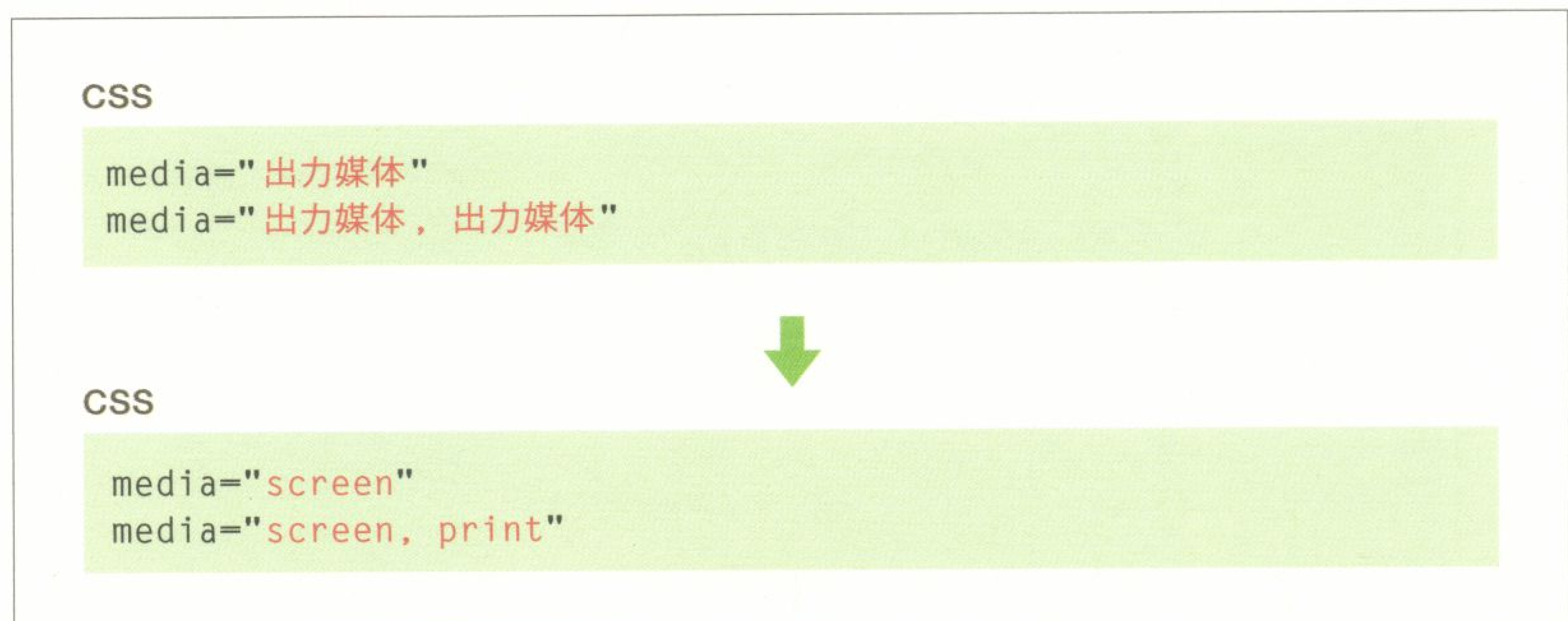
メディアクエリーは、Internet Explorerはバージョン9以降でしか対応していませんが、それ以外の比較的新しいブラウザではほぼ問題なく対応しています。

メディアクエリーの書き方 CSS3改

指定可能な出力媒体の特性（メディア特性）には右ページの表のものがあります。これらのメディア特性には、CSSのプロパティと同様の書式で値を指定して使います（値をつけずにメディア特性だけで指定できるものもあります）。値にはCSSで指定できるものと同じ単位が指定できます。たとえば、「min-width: 640px」のように書けば「表示領域の幅が640ピクセル以上」という意味になります。

▶▶ 出力媒体の指定

では、このような式を具体的にどう記述できるのかを説明していきましょう。CSS2.1でも指定可能だった出力媒体は、たとえばlink要素やstyle要素のmedia属性で次のように指定しました。



CSS

```
media="出力媒体"
media="出力媒体, 出力媒体"
```

CSS

```
media="screen"
media="screen, print"
```

CSS2.1で可能だった範囲のmedia属性の指定方法と指定例

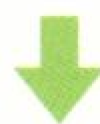
メディアクエリーを使用する場合は、出力媒体のあとに必要な数だけ「and （メディア特性: 値）」を追加して条件を加えていきます。すると、「出力媒体」と「メディア特性: 値」の式がすべて成り立つ場合にのみCSSが適用されることになります。もちろん、これまで同様にカンマで区切って複数の出力媒体（および and （メディア特性: 値））を指定することもできます。

メディア特性	説明	値
width min-width max-width	表示領域の幅　※スクロールバーも含む 表示領域の最小の幅（これ以上で適用） 表示領域の最大の幅（これ以下で適用）	実数＋単位 実数＋単位 実数＋単位
height min-height max-height	表示領域の高さ　※スクロールバーも含む 表示領域の最小の高さ（これ以上で適用） 表示領域の最大の高さ（これ以下で適用）	実数＋単位 実数＋単位 実数＋単位
device-width min-device-width max-device-width	出力機器の画面全体の幅 出力機器の画面全体の最小の幅（これ以上で適用） 出力機器の画面全体の最大の幅（これ以下で適用）	実数＋単位 実数＋単位 実数＋単位
device-height min-device-height max-device-height	出力機器の画面全体の高さ 出力機器の画面全体の最小の高さ（これ以上で適用） 出力機器の画面全体の最大の高さ（これ以下で適用）	実数＋単位 実数＋単位 実数＋単位
orientation	縦長・縦横同じ（portrait）／横長（landscape）	portrait, landscape
aspect-ratio min-aspect-ratio max-aspect-ratio	widthとheightの比率 widthとheightの最小の比率（これ以上で適用） widthとheightの最大の比率（これ以下で適用）	整数/整数　※例：4/3 整数/整数 整数/整数
device-aspect-ratio min-device-aspect-ratio max-device-aspect-ratio	device-widthとdevice-heightの比率 device-widthとdevice-heightの最小の比率（これ以上で適用） device-widthとdevice-heightの最大の比率（これ以下で適用）	整数/整数　※例：16/9 整数/整数 整数/整数
color min-color max-color	カラーコンポーネントのビット数 カラーコンポーネントのビット最小の数（これ以上で適用） カラーコンポーネントのビット最大の数（これ以下で適用）	整数 整数 整数
color-index min-color-index max-color-index	カラールックアップテーブルの色数 カラールックアップテーブルの最小の色数（これ以上で適用） カラールックアップテーブルの最大の色数（これ以下で適用）	整数 整数 整数
monochrome min-monochrome max-monochrome	モノクロのビット数 モノクロの最小のビット数（これ以上で適用） モノクロの最大のビット数（これ以下で適用）	整数 整数 整数
resolution min-resolution max-resolution	解像度 最小の解像度（これ以上で適用） 最大の解像度（これ以下で適用）	実数＋dpi, 実数＋dpcm 実数＋dpi, 実数＋dpcm 実数＋dpi, 実数＋dpcm
scan	テレビの走査方式	progressive, interlace
grid	ビットマップではないグリッド方式（文字幅固定の機器など）	整数

メディアクエリーで用意されているメディア特性

CSS

```
media="出力媒体 and （メディア特性： 値）"
media="出力媒体 and （メディア特性： 値） and （メディア特性： 値）"
media="出力媒体 and （メディア特性： 値） and （メディア特性： 値）, 出力媒体"
```



CSS

```
media="screen and (min-width: 640px)"
media="screen and (min-width: 640px) and (max-width: 800px)"
media="screen, and (min-width: 640px) and (max-width: 800px), print"
```

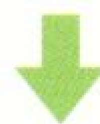
メディアクエリーを使用する場合の書式と指定例

@mediaについて

@media を使った書式を利用すると、CSSのソースコードの中に出力媒体とメディアクエリーの指定を書き込むことができます。こうすることで、特定の出力媒体が特定の状態になっている場合にのみ、その部分の指定を適用させることが可能になります。

CSS

```
@media 出力媒体 メディアクエリー {
  セレクタ { プロパティ： 値；・・・ }
  セレクタ { プロパティ： 値；・・・ }
  セレクタ { プロパティ： 値；・・・ }
  ・・・
}
```



CSS

```
@media screen and (min-width: 640px) and (max-width: 800px) {
  #main { float: none; }
  #sub { float: none; }
  p { font-size: 13px; }
}
```

@mediaの書式と指定例。この例では、パソコン画面で表示領域の幅が640ピクセル以上800ピクセル以下の場合にのみ、{ }内のCSSが適用される

CHAPTER 11

変形とアニメーション

CSS3では、ボックスを移動できるだけでなく、同時に拡大縮小や回転などもおこなえるようになっています。しかも、その変化の過程をアニメーションのように表示させることも可能です。現時点ではまだ対応ブラウザは限られていますが、CSS3の醍醐味とも言える機能をここで体験してみましょう。

LECTURE 11-1

トランスフォーム関連プロパティ

まずは、回転・拡大縮小・移動・傾斜についてです。これらはすべてtransformプロパティを使用しておこないます。これらの変形をおこなう際には、基準となる原点がどこにあるのかを考えることも重要です。

回転・拡大縮小・移動・傾斜 【CSS3新】

transformプロパティを使用すると、それだけで回転・拡大縮小・移動・傾斜のすべてを、しかも必要であれば同時におこなうことができます。ただし、このプロパティだけでは、変化の過程をアニメーションのように表示させることはできません。このプロパティで表示できるのは、最終的な表示結果だけです。

値はそれぞれの機能に合わせた関数形式になっており、半角スペースで区切って必要なだけ指定することができます。値は指定された順番に実行されますので、指定する順序によって表示結果が変わる場合もある点に注意してください。なお、このプロパティでボックスの大きさや位置などを変更しても、まわりの要素の配置位置には影響は一切ありません。指定できるのは次の値です(実際には3D用の値なども用意されていますが、本書では2D用の値を中心に抜粋して紹介しています)。

transformに指定できる値

- `rotate(角度)`
  時計回りに回転させる角度を単位(度=deg)つきの実数で指定します。

- `scale(実数, 実数)`
  拡大縮小させる倍率を横方向・縦方向の順に、単位をつけない実数でカンマで区切って指定します。値を1つだけ指定すると、その値が横方向・縦方向の両方に適用されます。

- `scaleX(実数)`
  横方向に拡大縮小させる倍率を単位をつけない実数で指定します。

- `scaleY(実数)`
  縦方向に拡大縮小させる倍率を単位をつけない実数で指定します。

transformに指定できる値(続き)

・**translate(単位つきの実数, 単位つきの実数)**
移動させる距離を右方向・下方向の順に、単位つきの実数またはパーセンテージで指定します。負の数値を指定すると左方向・上方向に移動します。値を1つだけ指定すると、右方向への移動距離だけを指定したことにります。

・**translateX(単位つきの実数)**
右方向に移動させる距離を単位つきの実数またはパーセンテージで指定します。負の数値を指定すると左方向に移動します。

・**translateY(単位つきの実数)**
下方向に移動させる距離を単位つきの実数またはパーセンテージで指定します。負の数値を指定すると上方向に移動します。

・**skew(角度)**
傾斜させる角度を、x軸に沿った傾斜の角度・y軸に沿った傾斜の角度の順に、単位(度=deg)つきの実数でカンマで区切って指定します。値を1つだけ指定すると、x軸に沿った傾斜の角度だけを指定したことになります。

・**skewX(角度)**
x軸に沿った傾斜をさせる角度を、単位(度=deg)つきの実数で指定します。

・**skewY(角度)**
y軸に沿った傾斜をさせる角度を、単位(度=deg)つきの実数で指定します。

・**none**
回転・拡大縮小・移動・傾斜を一切していない状態となります。

▶▶ transformプロパティの使用例

では使用例を見て、実際の機能を確認してみましょう。以下のサンプルでは、正方形のトマトの写真を4つ配置しています(sample1〜sample4のidが指定されています)。最初の写真はそのまま表示させ、2つめの写真は時計回りに45度回転させています。3つめの写真は、横方向にのみ3倍に拡大してあります。4つめの写真は、右に300ピクセル、上に300ピクセル移動させ、x軸に沿って45度傾斜させています。

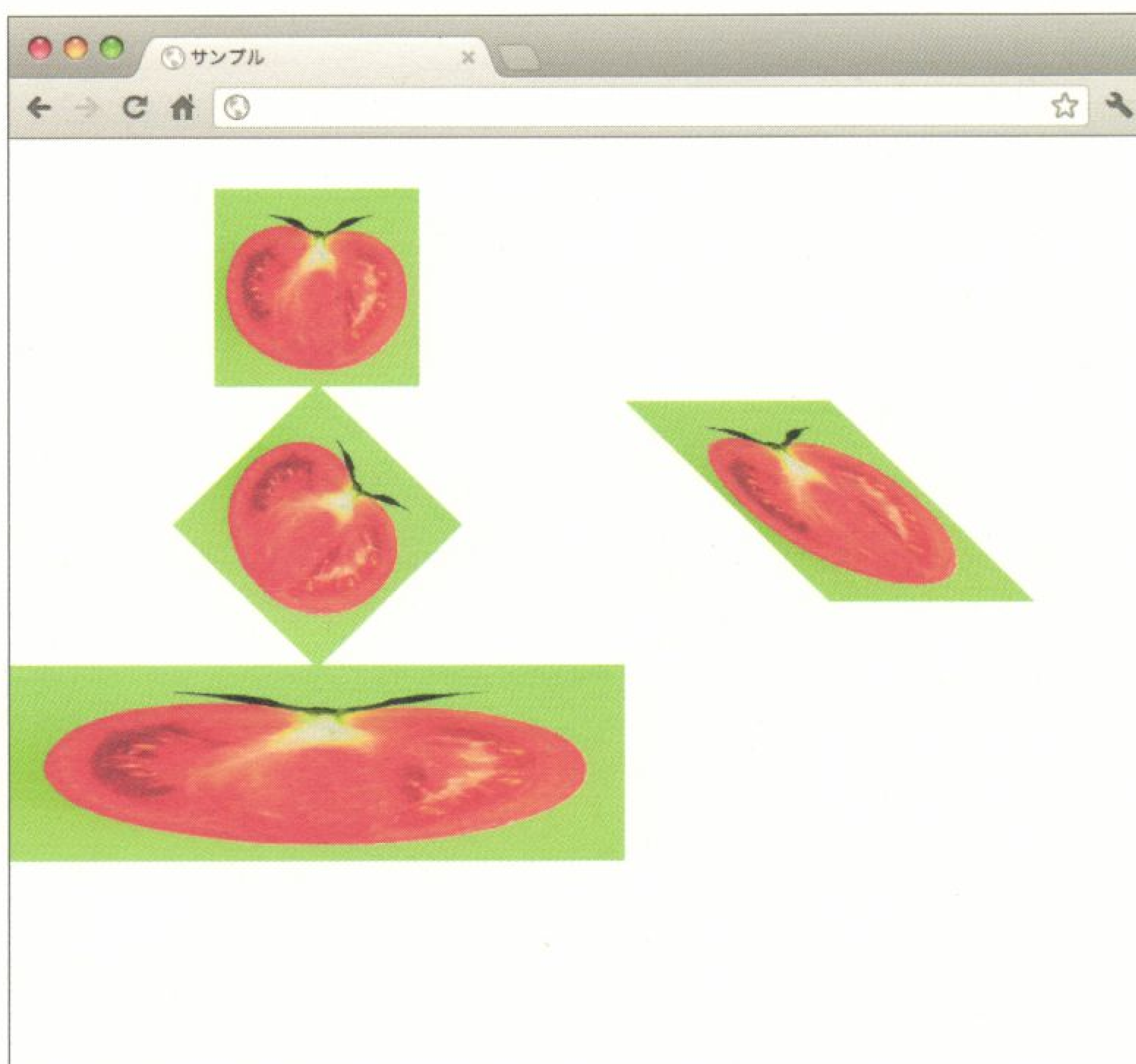
HTML　sample/chapter-11/lecture-11-1/01.html

```
<p><img id="sample1" src="tomato.jpg" alt="トマトの写真"></p>
<p><img id="sample2" src="tomato.jpg" alt="トマトの写真"></p>
<p><img id="sample3" src="tomato.jpg" alt="トマトの写真"></p>
<p><img id="sample4" src="tomato.jpg" alt="トマトの写真"></p>
```

CSS　sample/chapter-11/lecture-11-1/01-transform.css

```
body { margin: 30px 120px; }
img {
  width: 120px;
  height: 120px;
}
#sample2 {
  -webkit-transform: rotate(45deg);
  -moz-transform: rotate(45deg);
  -ms-transform: rotate(45deg);
  -o-transform: rotate(45deg);
  transform: rotate(45deg);
}
#sample3 {
  -webkit-transform: scaleX(3);
  -moz-transform: scaleX(3);
  -ms-transform: scaleX(3);
  -o-transform: scaleX(3);
  transform: scaleX(3);
}
#sample4 {
  -webkit-transform: translate(300px, -300px) skew(45deg);
  -moz-transform: translate(300px, -300px) skew(45deg);
  -ms-transform: translate(300px, -300px) skew(45deg);
  -o-transform: translate(300px, -300px) skew(45deg);
  transform: translate(300px, -300px) skew(45deg);
}
```

transformプロパティの使用例。4つの写真を表示させ、2つめから4つめをそれぞれ回転・拡大・移動と傾斜させている



上のソースコードを表示させたところ

このプロパティには、Internet Explorerもバージョン9から対応しています。現時点でこのプロパティを使用するには、各ブラウザのベンダープレフィックスをつける必要があります。

transformの原点の変更 CSS3新

transformプロパティのサンプルの表示結果を見ると、回転や拡大縮小などが、ボックスの中心を原点としておこなわれていることが分かります。この原点の位置を変更するには、**transform-originプロパティ**を使用します。次の値が指定できます。

transform-originに指定できる値

- **単位つきの実数**
  原点とする位置を単位つきの実数で指定します。
- **パーセンテージ**
  原点とする位置をパーセンテージで指定します。
- **top, right, bottom, left, center**
  原点とする位置をキーワードで指定します。

値は、ボックスの左上を0とする座標かキーワードで指定します。単位つきの実数またはパーセンテージで指定する場合は、x軸の値、y軸の値の順に半角スペースで区切って指定します。キーワードの場合は順序に関係なく半角スペースで区切って指定しできます。

実は、このプロパティの初期値は「50% 50%」であるため、transformプロパティのサンプルでは、回転や拡大縮小などがボックスの中心を原点としておこなわれていたというわけです。

▶▶ 原点を変更する

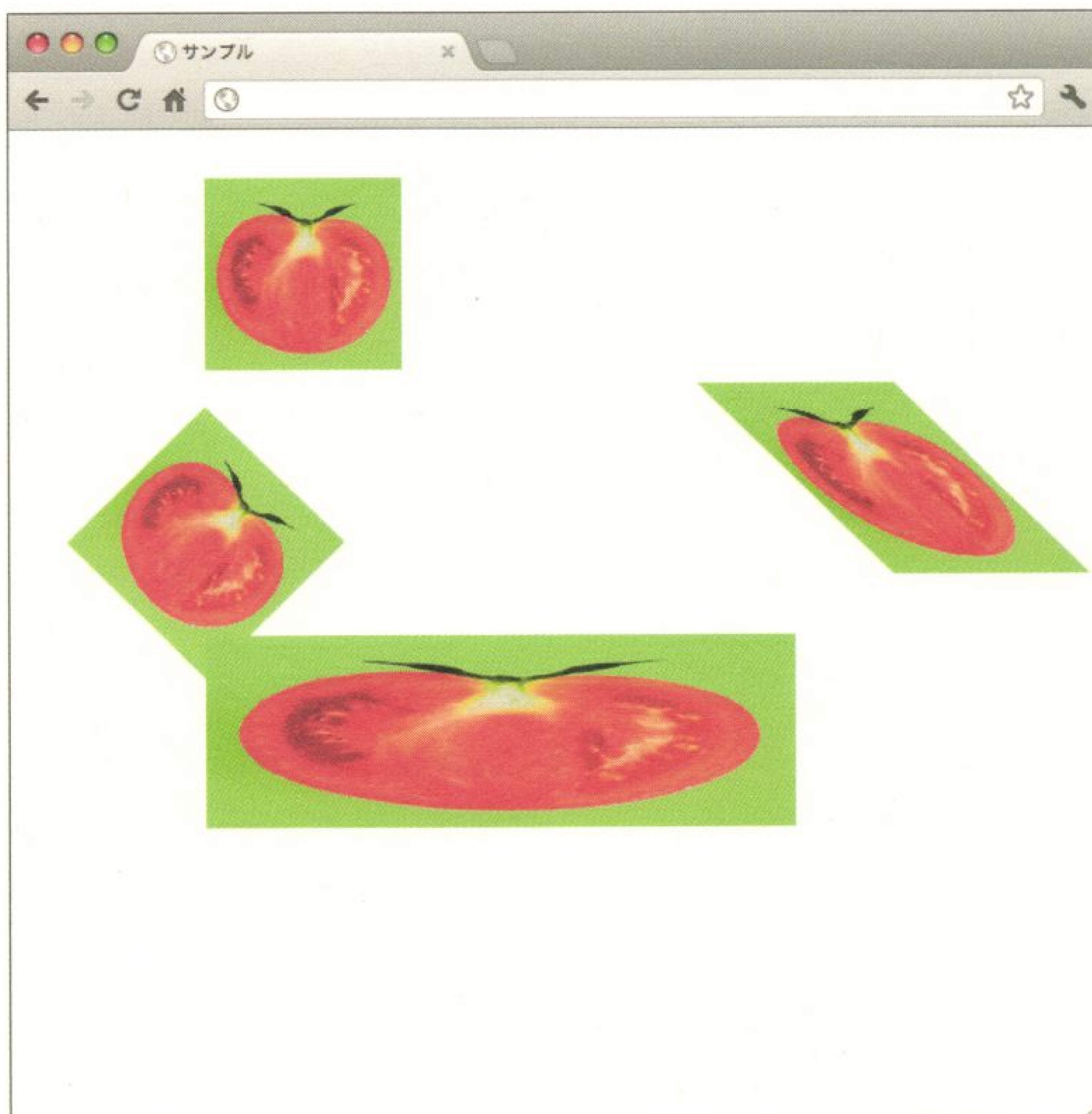
先程のサンプルに、ボックスの左上を原点にする指定を追加したものが次のサンプルです。表示結果を見ると、回転や拡大などが画像の左上を原点としておこなわれていることが分かります。

CSS　sample/chapter-11/lecture-11-1/02-transform-origin.css（抜粋）

```
body { margin: 30px 120px; }
img {
  width: 120px;
  height: 120px;
  -webkit-transform-origin: top left;
  -moz-transform-origin: top left;
  -ms-transform-origin: top left;
  -o-transform-origin: top left;
  transform-origin: top left;
}
・・・
```

追加（05〜09行）

原点をボックスの左上に変更するソースを追加



回転や拡大などが、画像の左上を原点としておこなわれるようになった

LECTURE 11-2

トランジション関連プロパティ

徐々に変化させる効果を指定できるのがCSS3のトランジション関連プロパティです。

トランジションの基本プロパティ 【CSS3新】

セレクタの :hover を使用するなどして、あるプロパティの値が変化するときに、瞬時に切り替わるのではなく指定した時間をかけてアニメーションのように徐々に変化させるのがCSS3のトランジションです。どのプロパティの値が変化したときにトランジションを適用するのかを指定するのが**transition-property プロパティ**で、変化にかける時間は**transition-duration プロパティ**で指定します。それぞれ次の値が指定できます。

transition-propertyに指定できる値

- **プロパティ名**
  トランジションを適用するプロパティの名前をそのまま指定します。カンマで区切って複数指定できます。

- `all`
  トランジションの適用が可能なすべてのプロパティに適用します。

- `none`
  トランジションを適用しません。

transition-durationに指定できる値

- **時間**
  トランジションの変化にかける時間を単位付きの数値で指定します。単位には、「s（秒）」と「ms（ミリ秒）」が指定できます。

▶▶ transition-propertyとtransition-durationの使用例

では、具体的にどうなるのかをサンプルで見てみましょう。HTMLではサメのおもちゃの写真（背景を透明にした透過PNG）を表示させているだけです。CSSでは、まずbody要素の背景として海底の写真を表示させています。そして、transition-propertyプロパティで、transformプロパティの値が変化したときにトランジションを適用するように指定しています。このとき、現時点ではtransition-propertyとtransformの両方にベンダープレフィックスの指定が必要である点に注意してください。そして同じくベンダープレフィックスをつけたtransition-durationプロパティで、トランジションにかける時間を2秒（2s）に設定しています。CSSの最後に、画像の上にカーソルを重ねたときの指定があり、そこではtransformプロパティを使用してサメの写真を右に300ピクセル移動させています。では、このサンプルをブラウザで表示させて、サメの写真の上にカーソルをのせてみてください。

HTML　　sample/chapter-11/lecture-11-2/01.html

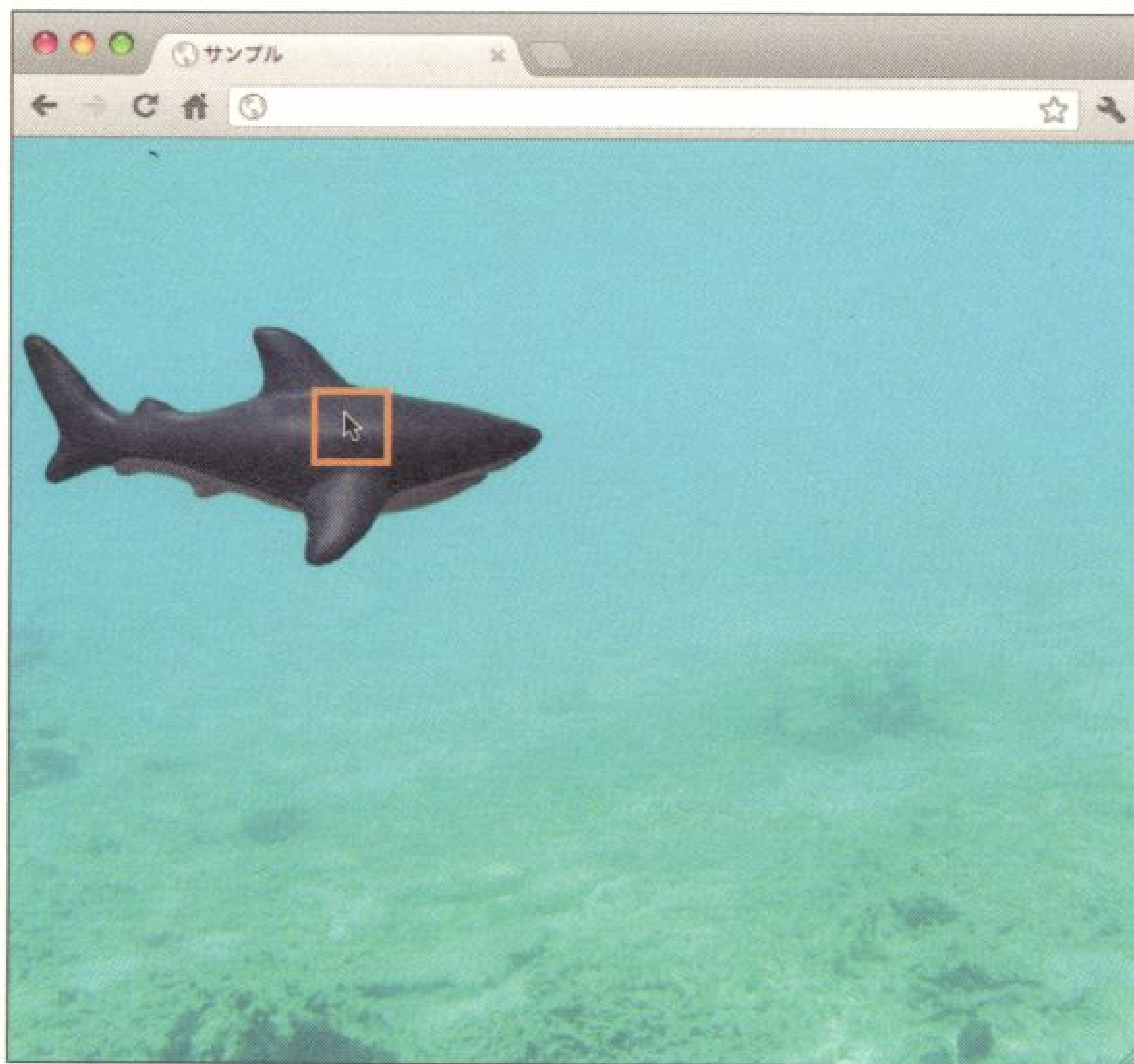
```
<div>
<img src="shark.png" alt="サメの写真">
</div>
```

CSS　　sample/chapter-11/lecture-11-2/01-transition-property.css

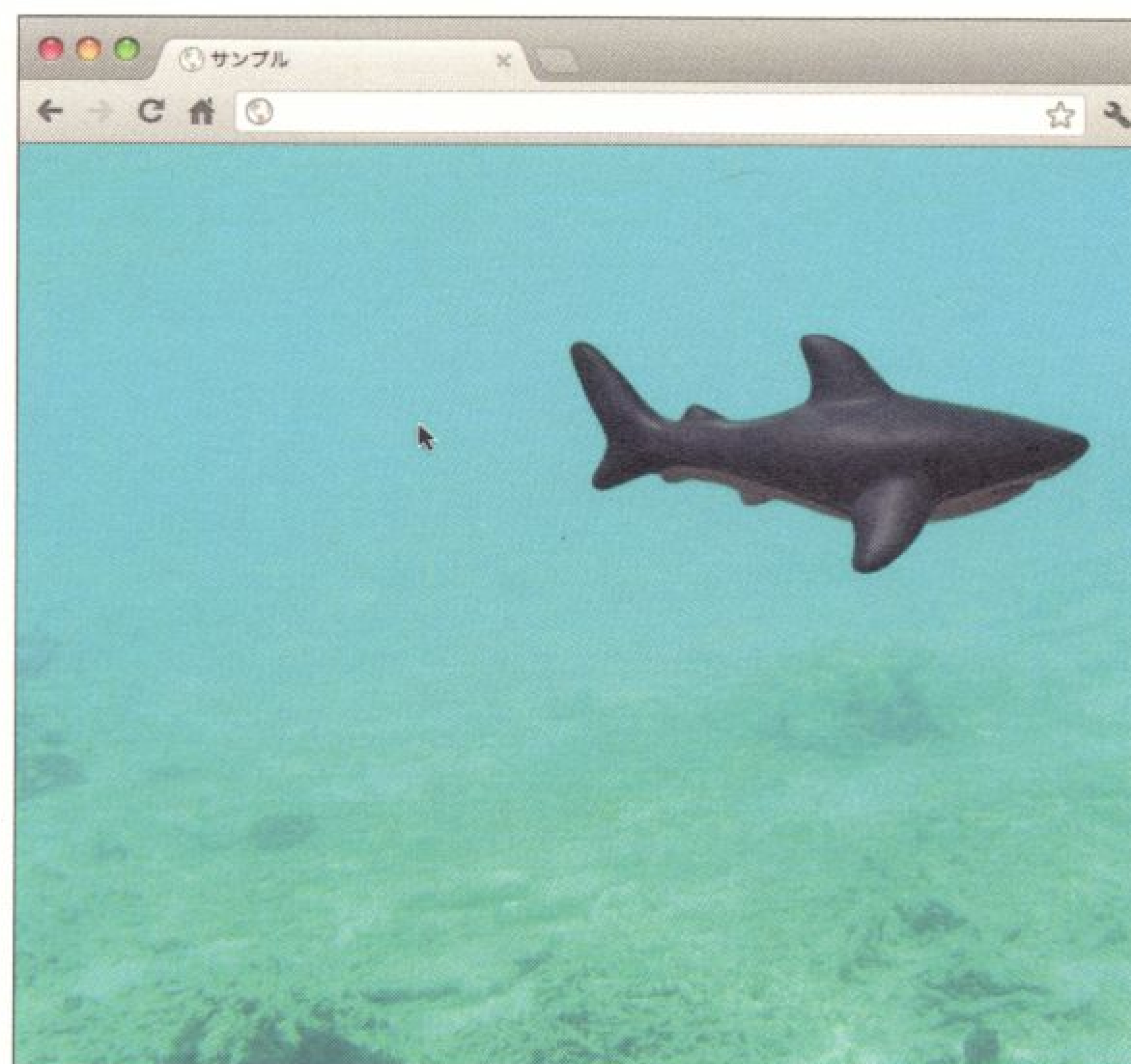
```
html, body { height: 100%; }
body {
  margin: 0;
  background: url(sea.jpg);
  -webkit-background-size: cover;
  -moz-background-size: cover;
  background-size: cover;
}
img {
  margin-top: 100px;
  width: 300px;
  height: 150px;
  -webkit-transition-property: -webkit-transform;
  -webkit-transition-duration: 2s;
  -moz-transition-property: -moz-transform;
  -moz-transition-duration: 2s;
  -ms-transition-property: -ms-transform;
  -ms-transition-duration: 2s;
  -o-transition-property: -o-transform;
  -o-transition-duration: 2s;
  transition-property: transform;
  transition-duration: 2s;
}
img:hover {
  -webkit-transform: translateX(300px);
  -moz-transform: translateX(300px);
  -ms-transform: translateX(300px);
```

```
  -o-transform: translateX(300px);
  transform: translateX(300px);
}
```

トランジションのサンプルのソースコード



①サメの写真の上にカーソルをのせる



②すると、サメはなめらかに右側に移動する

このサンプルでは場所を移動させていますが、transformプロパティが使用できるということはもちろん回転や拡大縮小もできますし、それ以外のcolorプロパティやbackgroundプロパティで色などを変えることも可能です。
サンプルでは4種類のブラウザ向けのベンダープレフィックスを指定してはいますが、Interenet Explorerについてはバージョン9でも未対応です。しかしバージョン10では対応予定となっていますので、Interenet Explorer向けのベンダープレフィックスも加えてあります。

トランジションのその他のプロパティ 【CSS3新】

transition-delayプロパティは、トランジションの開始を遅らせるためのプロパティです。初期値は0s(0秒)で、指定できるのは遅らせる時間だけです。

transition-delayに指定できる値

- **時間**
  変化を開始するまでの時間を単位付きの数値で指定します。単位には、「s(秒)」と「ms(ミリ秒)」が指定できます。

トランジションの開始から終了までの総合時間はtransition-durationプロパティで指定したもので固定ですが、その時間内での加速や減速の加減は**transition-timing-functionプロパティ**で調整することができます。次の値が指定できます。初期値は「ease」です。

transition-timing-functionに指定できる値

- ease
  加速をつけて、ゆっくりと始まり、ゆっくりと終わります。
- ease-in
  ゆっくりと始まり、一定速度で終わります。
- ease-out
  一定速度で始まり、ゆっくりと終わります。
- ease-in-out
  ゆっくりと始まり、ゆっくりと終わります。
- linear
  最初から最後まで一定の速度で変化します。

▶▶transition-timing-functionの使用例

では、transition-timing-functionプロパティによる変化速度の違いをサンプルで見てみましょう。このサンプルではサメの写真を縦に5つ表示させ、その上にカーソルをのせると動くようになっています。それぞれにtransition-timing-functionプロパティを指定して、加速や減速のタイミングを変えてあります。なお、すべてのベンダープレフィックスを記述しているとソースコードが大変長くなってしまい、指定内容が理解しにくくなってしまいますので、このサンプルでは「-webkit-」向けの指定のみ記述しています。

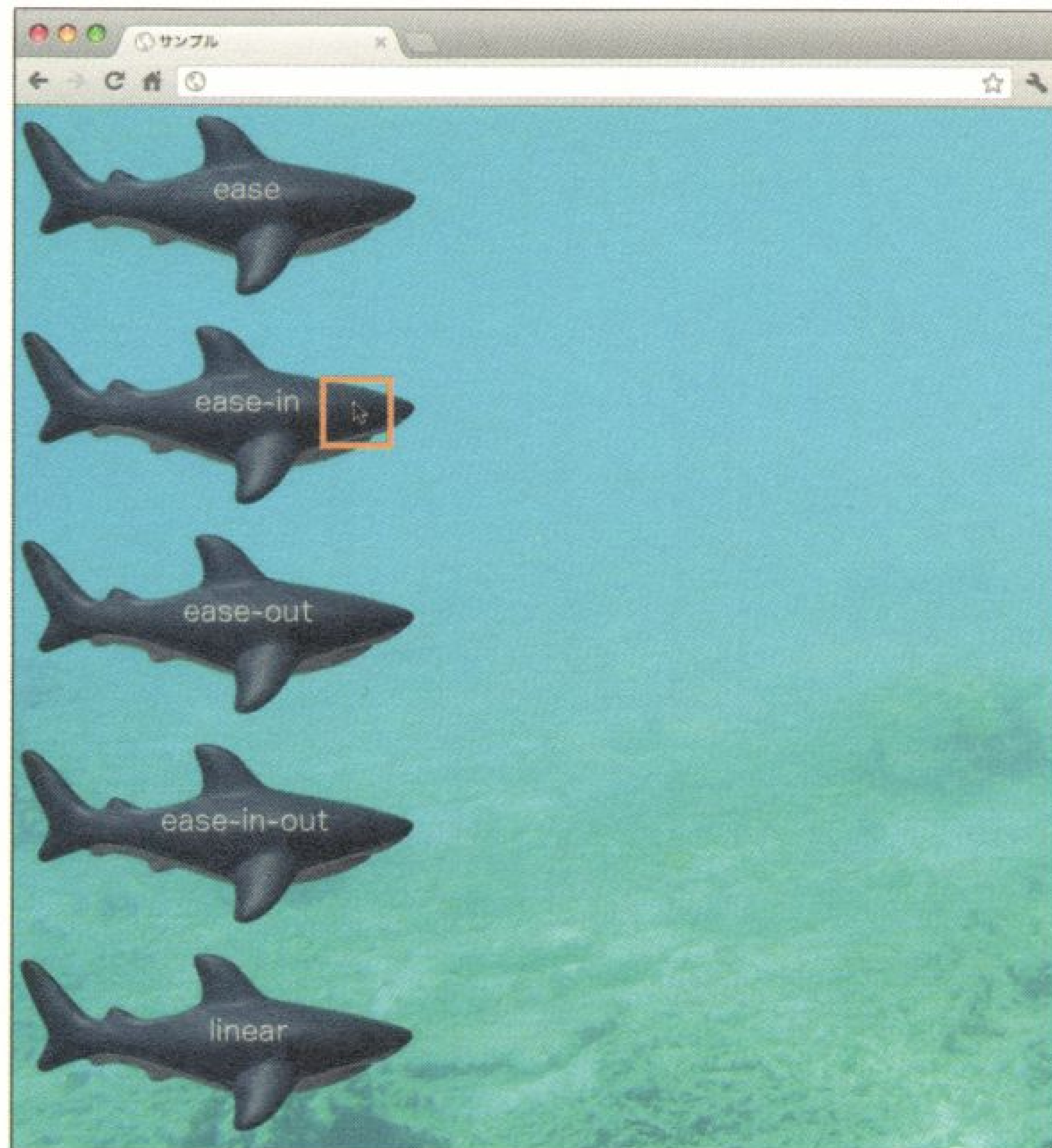
HTML　sample/chapter-11/lecture-11-2/02.html

```
<div>
<img id="sample1" src="shark1.png" alt="サメの写真"><br>
<img id="sample2" src="shark2.png" alt="サメの写真"><br>
<img id="sample3" src="shark3.png" alt="サメの写真"><br>
<img id="sample4" src="shark4.png" alt="サメの写真"><br>
<img id="sample5" src="shark5.png" alt="サメの写真">
</div>
```

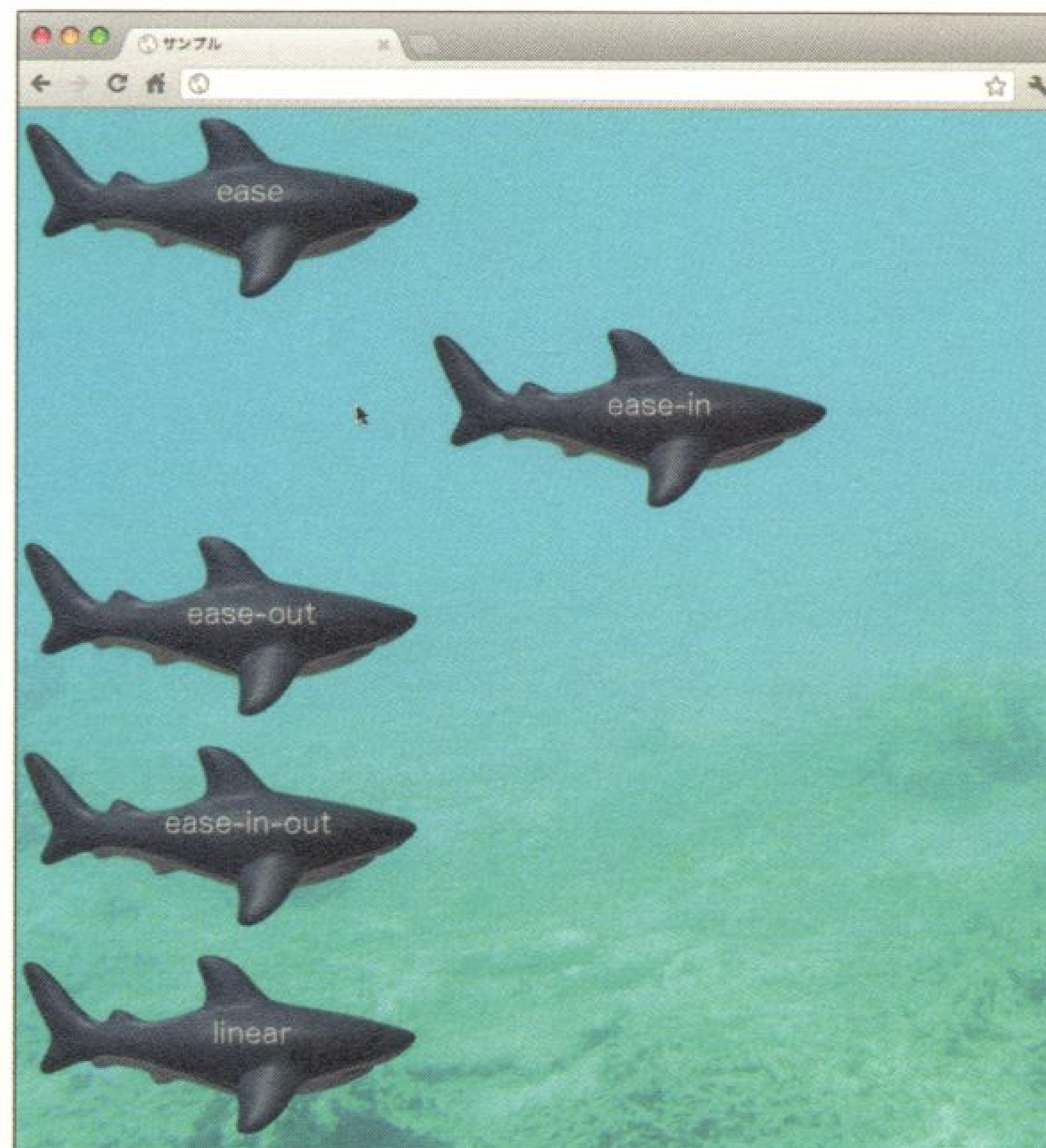
CSS　sample/chapter-11/lecture-11-2/02-timing-function.css

```css
html, body { height: 100%; }
body {
  margin: 0;
  background: url(sea.jpg);
  -webkit-background-size: cover;
  -moz-background-size: cover;
  background-size: cover;
}
img {
  width: 300px;
  height: 150px;
  -webkit-transition-property: -webkit-transform;
  -webkit-transition-duration: 2s;
}
img:hover { -webkit-transform: translateX(300px); }
#sample1 { -webkit-transition-timing-function: ease; }
#sample2 { -webkit-transition-timing-function: ease-in; }
#sample3 { -webkit-transition-timing-function: ease-out; }
#sample4 { -webkit-transition-timing-function: ease-in-out; }
#sample5 { -webkit-transition-timing-function: linear; }
```

transition-timing-functionプロパティにより、トランジション時の加速や減速のタイミングを変えたサンプル



①サンプルをブラウザで表示させたところ



②サメの写真にカーソルをのせると、指定された速度の加減で動き出す

このサンプルでは、すべての写真が同時に動くわけではないのでそれぞれの違いが比較しにくいかもしれません。このあとのアニメーションの解説に出てくる同様のサンプル（sample/chapter-11/lecture-11-3/02.html）ではすべての写真が同時に動きますので、速度変化の違いなどはあらためてそちらで確認してください。

トランジションの一括指定 【CSS3新】

transitionプロパティを使用すると、トランジション関連の値を半角スペースで区切って**一度に指定**することができます。現時点(2012年7月)では、最初に指定された時間はtransition-durationプロパティの値で、2番目に指定された数値はtransition-delayプロパティの値ということになっていますが、この仕様は最終的には変更される可能性もあります(時間を2つ指定するときはスラッシュで区切る案も出ています)。

transitionに指定できる値

- **transition-propertyの値**
  transition-propertyプロパティに指定できる値が指定できます。
- **transition-durationの値**
  transition-durationプロパティに指定できる値が指定できます。
- **transition-timing-functionの値**
  transition-timing-functionプロパティに指定できる値が指定できます。
- **transition-delayの値**
  transition-delayプロパティに指定できる値が指定できます。

次のサンプルは、transitionプロパティを使用したものです。サメの写真にカーソルをのせると、サメが右に移動しつつ0.2倍に縮小されます。

HTML　sample/chapter-11/lecture-11-2/03.html

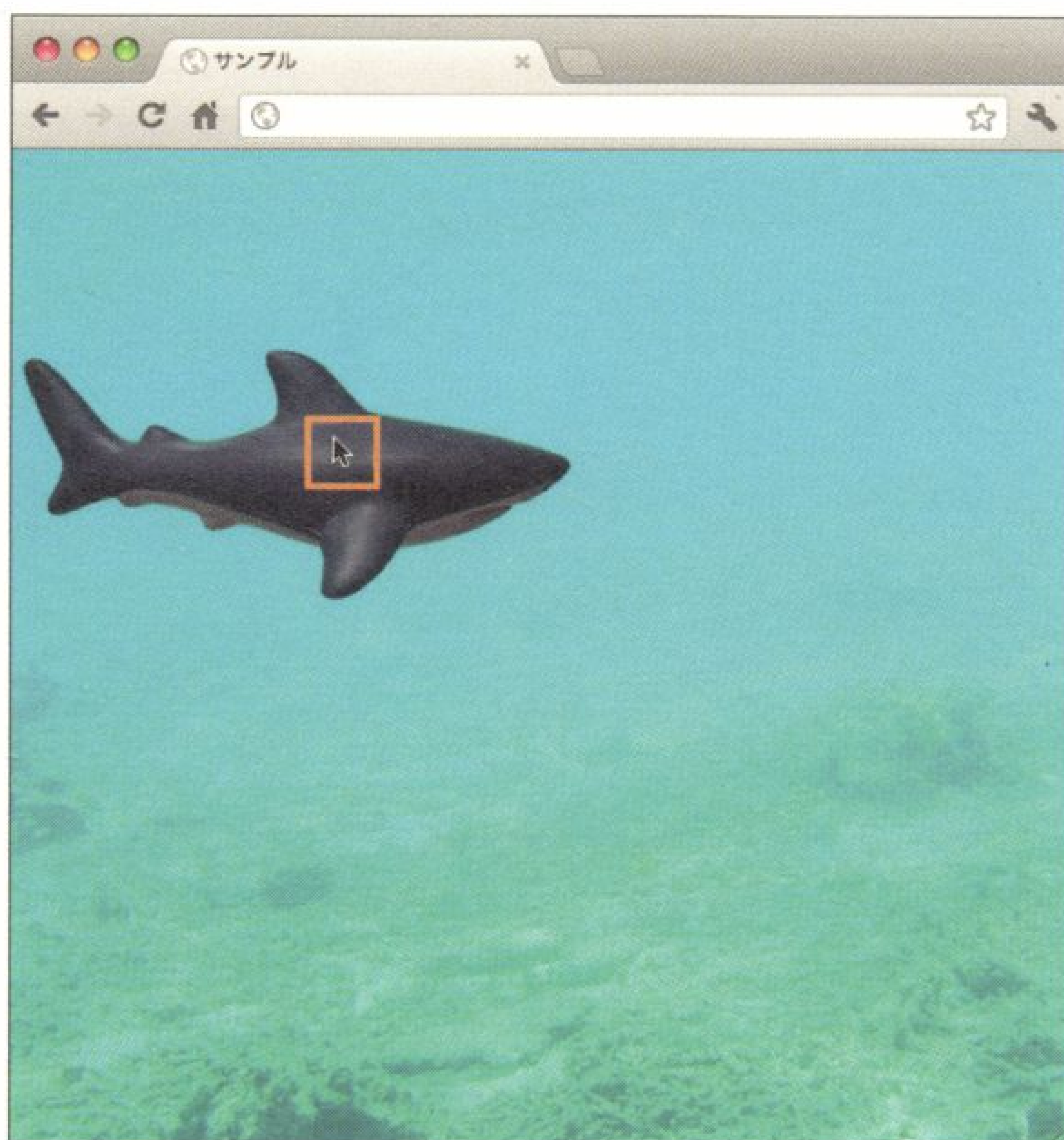
```
<div>
<img src="shark.png" alt="サメの写真">
</div>
```

CSS　sample/chapter-11/lecture-11-2/03-transition.css

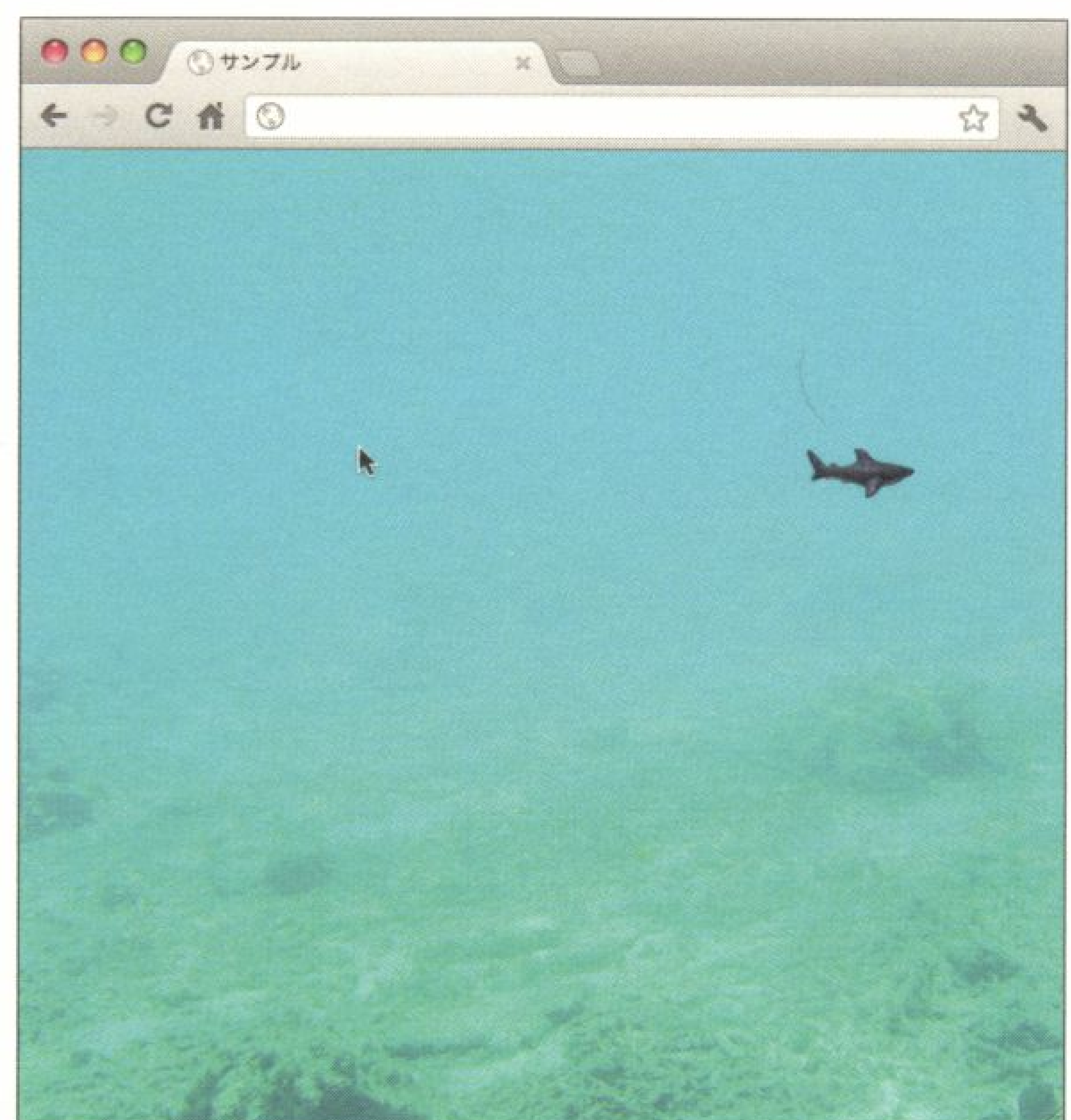
```
html, body { height: 100%; }
body {
  margin: 0;
  background: url(sea.jpg);
  -webkit-background-size: cover;
  -moz-background-size: cover;
  background-size: cover;
}
img {
  margin-top: 100px;
```

```
  width: 300px;
  height: 150px;
  -webkit-transition: -webkit-transform 2s;
  -moz-transition: -moz-transform 2s;
  -ms-transition: -ms-transform 2s;
  -o-transition: -o-transform 2s;
  transition: transform 2s;
}
img:hover {
  -webkit-transform: translateX(300px) scale(0.2);
  -moz-transform: translateX(300px) scale(0.2);
  -ms-transform: translateX(300px) scale(0.2);
  -o-transform: translateX(300px) scale(0.2);
  transform: translateX(300px) scale(0.2);
}
```

トランジションをtransitionプロパティだけで指定している例



①上のソースコードをブラウザで表示させたところ



②サメの写真にカーソルをのせると、右に移動しつつ小さくなっていく

LECTURE 11-3

アニメーション関連プロパティ

アニメーション関連プロパティを使うと、アニメーション全体で、プロパティの状態を細かく設定することができます。

@keyframesの指定方法 【CSS3新】

CSS3のアニメーションとは、簡単に言えばトランジションを連続して実行させるようなものです。アニメーション全体の時間の中のどの時点でどのプロパティがどの状態になるのかということを@keyframes { }という書式で指定して実行させます。

```
@keyframes 名前 {
  0% {
    プロパティ: 値;
    プロパティ: 値;
    ・・・
  }
  ○% {
    プロパティ: 値;
    プロパティ: 値;
    ・・・
  }

  ・・・

  ○% {
    プロパティ: 値;
    プロパティ: 値;
    ・・・
  }
  100% {
    プロパティ: 値;
    プロパティ: 値;
    ・・・
  }
}
```

CSS3のアニメーションの@keyframesの書式

はじめに「@keyframes」と書き、半角スペースで区切ってこの書式（キーフレーム）に名前をつけます。2012年7月現在ではこの「@keyframes」にもベンダープレフィックスをつける必要があり、「@-webkit-keyframes」や「@-moz-keyframes」のように書かなければ動作しません。

%の指定については、アニメーション全体の時間の中の何%の時点で、どのプロパティの値がどの状態なるのかを記述していきます。プロパティと値はいくつでも記述できます。0% { }と 100% { }の指定は必須ですが、そのあいだの%は自由に指定できます。

なお、このような指定をベンダープレフィックス別に記入していくと、ソースコードが大変長くなってしまい、全体の内容を把握するのがむずかしくなってしまいます。そのため、本書では「-webkit-」をつけた指定のみを記入して、その他のブラウザ向けの指定は以降すべて省略することにします。IE9もOpera11もアニメーションには未対応ですが、Firefoxはバージョン5から対応していますので、「-moz-」をつけた指定も追加するとFirefoxでも動作するようになります。

アニメーションの基本プロパティ 【CSS3新】

アニメーションを実行させるには、実行させたい「@keyframes」の名前を**animation-nameプロパティ**の値として指定します。カンマで区切って複数の名前を指定することもできます。
「@keyframes」で指定したアニメーション全体の長さ（時間）は、**animation-durationプロパティ**で指定します。それぞれのプロパティに指定できるのは、次の値です。

animation-nameに指定できる値

- **キーフレーム名**
  @keyframes ○○○ { } で指定したキーフレームの名前（○○○）を指定します。

- **none**
  アニメーションを実行しません。

animation-durationに指定できる値

- **時間**
  アニメーションにかける時間を単位付きの数値で指定します。単位には、「s（秒）」と「ms（ミリ秒）」が指定できます。

CHAPTER 11

▶▶ アニメーションの実例

では、実際にどうやってアニメーションさせるのかをサンプルで見てみましょう。次のサンプルでは、「@keyframes」に「shark」という名前をつけています。CSSのソースコードの最後の4行のところで、:hoverでそれを実行させています。アニメーションのトータル時間は、animation-durationプロパティで20秒に設定しています。

HTML　**sample/chapter-11/lecture-11-3/01.html**

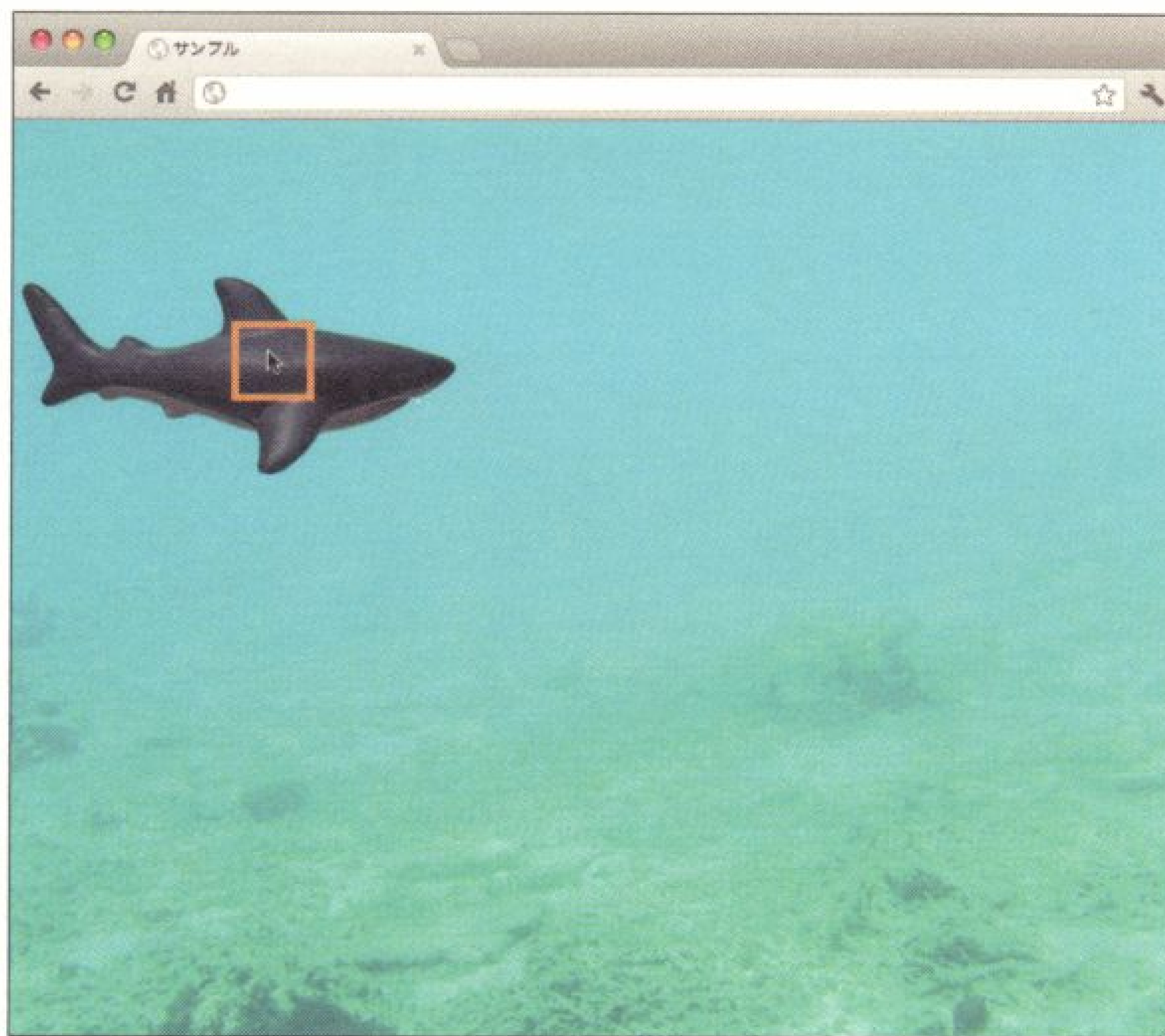
```
<div>
<img src="shark.png" alt="サメの写真">
</div
```

CSS　**sample/chapter-11/lecture-11-3/01-animation-name.css**

```
html, body { height: 100%; }
body {
  margin: 0;
  background: url(sea.jpg);
  -webkit-background-size: cover;
  -moz-background-size: cover;
  background-size: cover;
}
img {
  margin-top: 100px;
  width: 300px;
  height: 150px;
}
@-webkit-keyframes shark {
  0% {
    -webkit-transform: scale(1.0);
  }
  20% {
    opacity: 0.6;
    -webkit-transform: translateX(500px) scale(0.2);
  }
  21% {
    opacity: 0.5;
    -webkit-transform: translateX(500px) rotate(-90deg) scale(0.2);
  }
  22% {
    opacity: 0.5;
      -webkit-transform: translate(500px, -500px) rotate(-90deg)
scale(0.2);
  }
  23% {
    -webkit-transform: translate(-500px, -500px);
  }
  24% {
    -webkit-transform: translate(-500px);
  }
```

```
  98% {
    -webkit-transform: translate(-300px) scale(6.0);
  }
  100% {
    -webkit-transform: scale(1.0);
  }
}
img:hover {
  -webkit-animation-name: shark;
  -webkit-animation-duration: 20s;
}
```

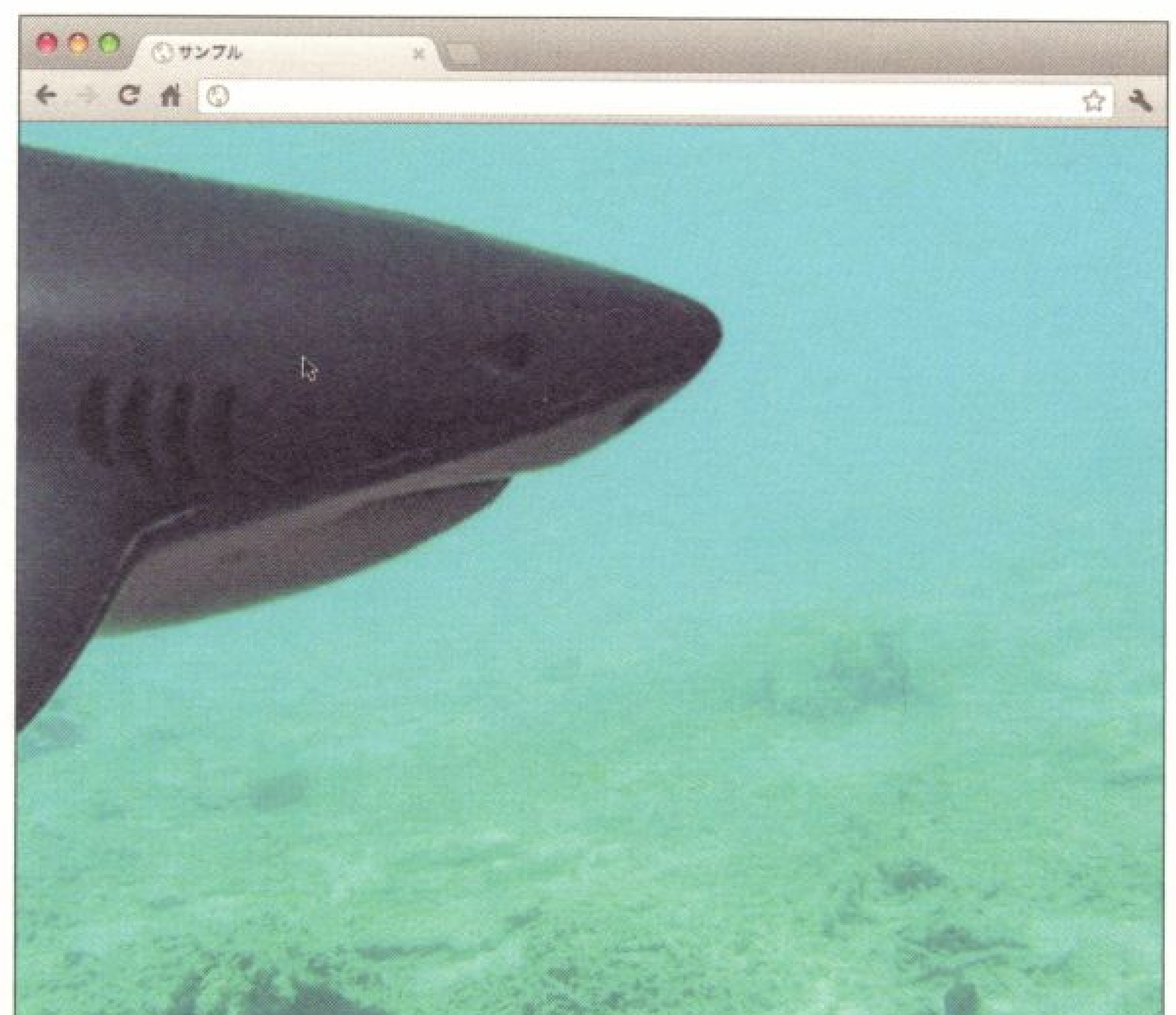
CSS3によるアニメーションのサンプルのソースコード



①サンプルをブラウザで表示させたところ



②サメの上にカーソルをのせると右に移動して小さくなると同時に半透明になり、サメは画面上に一旦消える



③その後、サメは画面左側からゆっくりと現れる

アニメーションのその他のプロパティ 【CSS3新】

トランジションには、開始を遅らせる**transition-delayプロパティ**と、加速や減速の加減を調整する**transition-timing-functionプロパティ**がありました。アニメーションにも、それらとまったく同じ機能で同じ値が指定できる**animation-delayプロパティ**と**animation-timing-functionプロパティ**が用意されています。

animation-delayに指定できる値

- **時間**
  アニメーションを開始するまでの時間を単位付きの数値で指定します。単位には、「s（秒）」と「ms（ミリ秒）」が指定できます。

animation-timing-functionに指定できる値

- **ease**
  加速をつけて、ゆっくりと始まり、ゆっくりと終わります。
- **ease-in**
  ゆっくりと始まり、一定速度で終わります。
- **ease-out**
  一定速度で始まり、ゆっくりと終わります。
- **ease-in-out**
  ゆっくりと始まり、ゆっくりと終わります。
- **linear**
  最初から最後まで一定の速度で変化します。

アニメーションには、トランジションにはない種類のプロパティも用意されています。それは、繰り返しに関するプロパティです。**animation-iteration-countプロパティ**は、アニメーションを繰り返す回数を指定するプロパティで、初期値は1になっています。値として「infinite」を指定すると、アニメーションを無限に繰り返します。

再生の際に、キーフレーム通りに再生するか逆再生するかを指定できるのが**animation-directionプロパティ**です。アニメーションを繰り返す際には、キーフレーム通りの再生と逆再生を順に繰り返したり、その逆をおこなうこともできます。初期値は「normal」で、常にキーフレーム通りに再生します。これらのプロパティに指定できるのは、次の値です。

animation-iteration-countに指定できる値

・**実数**
アニメーションを繰り返す回数を指定します。初期値は1です。

・**infinite**
アニメーションを無限に繰り返します。

animation-directionに指定できる値

・**normal**
常にキーフレーム通りに再生します。

・**reverse**
常に逆再生します。

・**alternate**
繰り返しの際、キーフレーム通りの再生と逆再生を順に繰り返します。

・**alternate-reverse**
繰り返しの際、逆再生とキーフレーム通りの再生を順に繰り返します。

▶▶ アニメーション関連プロパティの使用例

では、これらのプロパティを使用したサンプルを見てみましょう。ソースコードは次の通りです。transition-timing-functionプロパティのサンプルのアニメーション版で、サンプルをブラウザで表示させると、animation-delayプロパティによって2秒後にアニメーションが開始されます(このサンプルは :hoverで実行されるのではなくimg要素にCSSが適用されると同時にアニメーションが実行されます)。アニメーションはウィンドウを閉じるまで繰り返され、繰り返しの際はキーフレーム通りの再生と逆再生を順に繰り返します。

HTML　sample/chapter-11/lecture-11-3/02.html

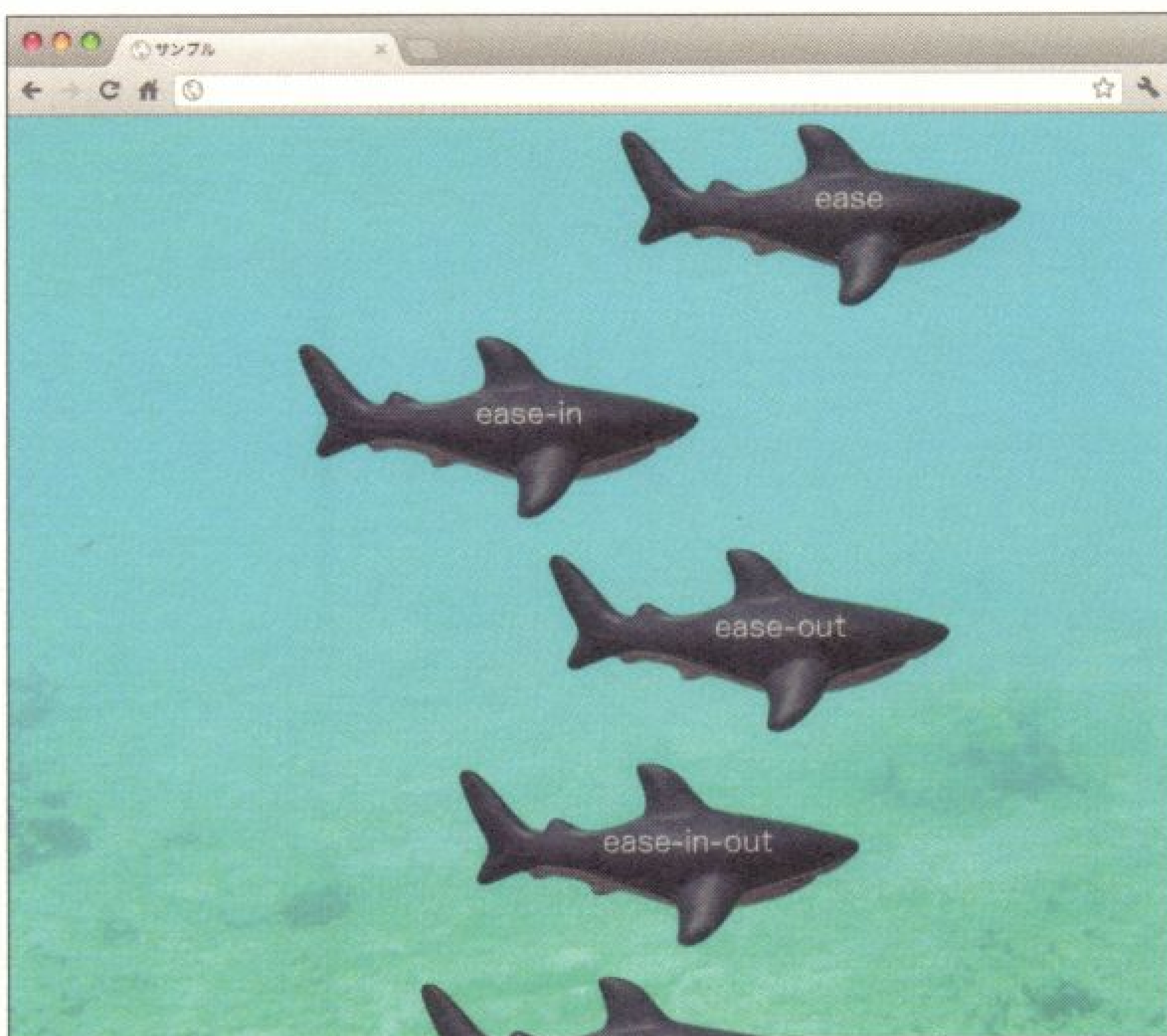
```
<div>
<img id="sample1" src="shark1.png" alt="サメの写真"><br>
<img id="sample2" src="shark2.png" alt="サメの写真"><br>
<img id="sample3" src="shark3.png" alt="サメの写真"><br>
<img id="sample4" src="shark4.png" alt="サメの写真"><br>
<img id="sample5" src="shark5.png" alt="サメの写真">
</div>
```

CHAPTER 11

CSS　sample/chapter-11/lecture-11-3/02-timing-function.css

```
html, body { height: 100%; }
body {
  margin: 0;
  background: url(sea.jpg);
  -webkit-background-size: cover;
  -moz-background-size: cover;
  background-size: cover;
}
@-webkit-keyframes shark {
  0% { -webkit-transform: translateX(0) }
  100% { -webkit-transform: translateX(500px) }
}
img {
  width: 300px;
  height: 150px;
  -webkit-animation-name: shark;
  -webkit-animation-duration: 2s;
  -webkit-animation-iteration-count: infinite;
  -webkit-animation-direction: alternate;
  -webkit-animation-delay: 2s;
}
#sample1 { -webkit-animation-timing-function: ease; }
#sample2 { -webkit-animation-timing-function: ease-in; }
#sample3 { -webkit-animation-timing-function: ease-out; }
#sample4 { -webkit-animation-timing-function: ease-in-out; }
#sample5 { -webkit-animation-timing-function: linear; }
```

アニメーション用の各種プロパティを使用したサンプル



このサンプルをブラウザで表示させると、何もしなくても2秒後にアニメーションが開始される

アニメーションの一括指定 【CSS3新】

トランジションと同様に、アニメーションにも値を一括指定できるプロパティが用意されています。アニメーション関連の値を半角スペースで区切って指定できます。トランジションの場合と同様、時間の値を指定した場合は1つめがanimation-durationプロパティとなり、2つめがanimation-delayプロパティの値となります(トランジション同様、この仕様は変更される可能性があります)。

animationに指定できる値

- **animation-nameの値**
  animation-nameプロパティに指定できる値が指定できます。
- **animation-durationの値**
  animation-durationプロパティに指定できる値が指定できます。
- **animation-timing-functionの値**
  animation-timing-functionプロパティに指定できる値が指定できます。
- **animation-iteration-countの値**
  animation-iteration-countプロパティに指定できる値が指定できます。
- **animation-directionの値**
  animation-directionプロパティに指定できる値が指定できます。
- **animation-delayの値**
  animation-delayプロパティに指定できる値が指定できます。

▶▶ animationプロパティの使用例

次のサンプルは、animationプロパティを使用したものです。アニメーションは、サンプルをブラウザで開くとすぐに開始されます。

HTML　sample/chapter-11/lecture-11-3/03.html

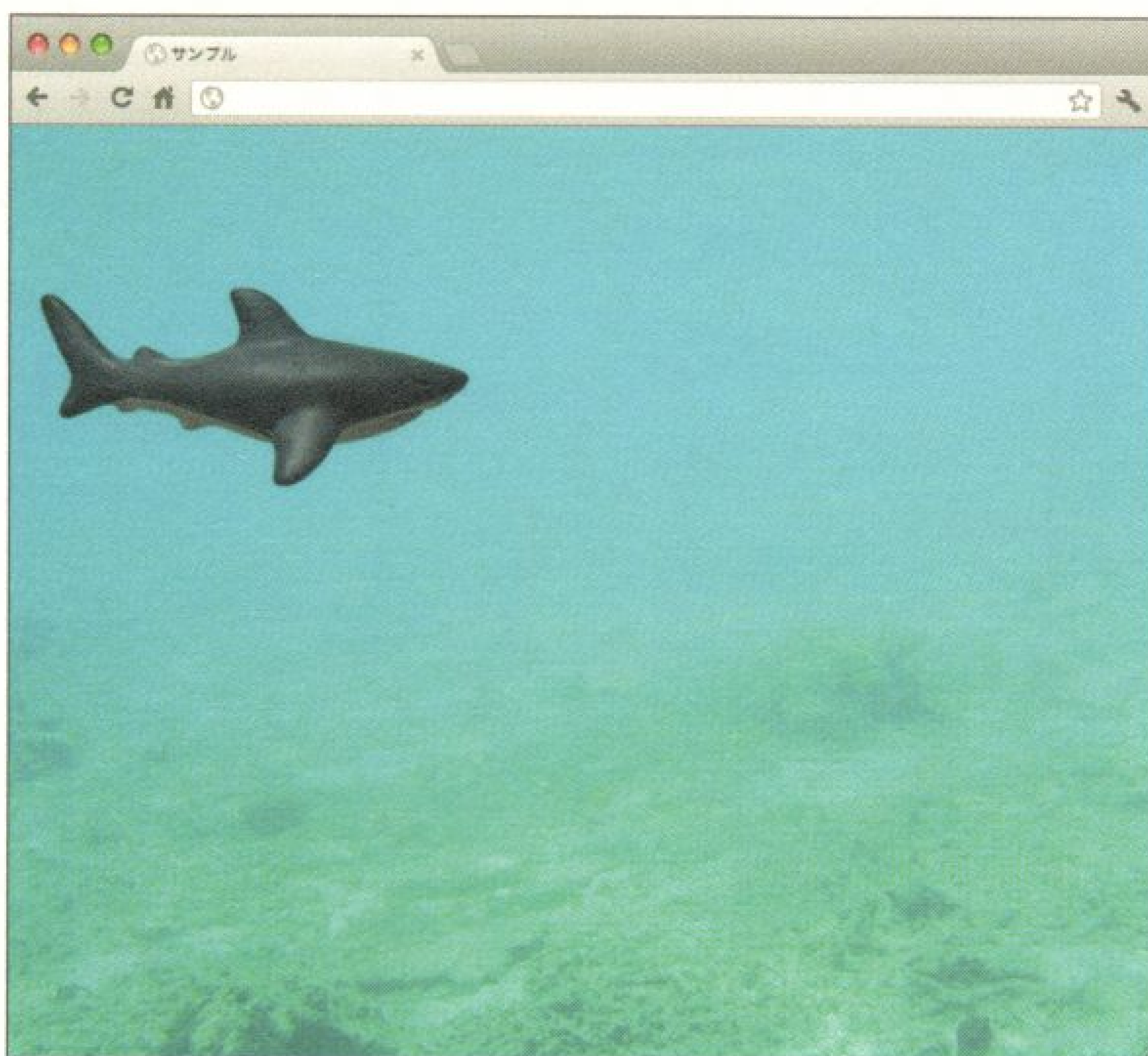
```
<div>
<img src="shark.png" alt="サメの写真">
</div>
```

CHAPTER 11

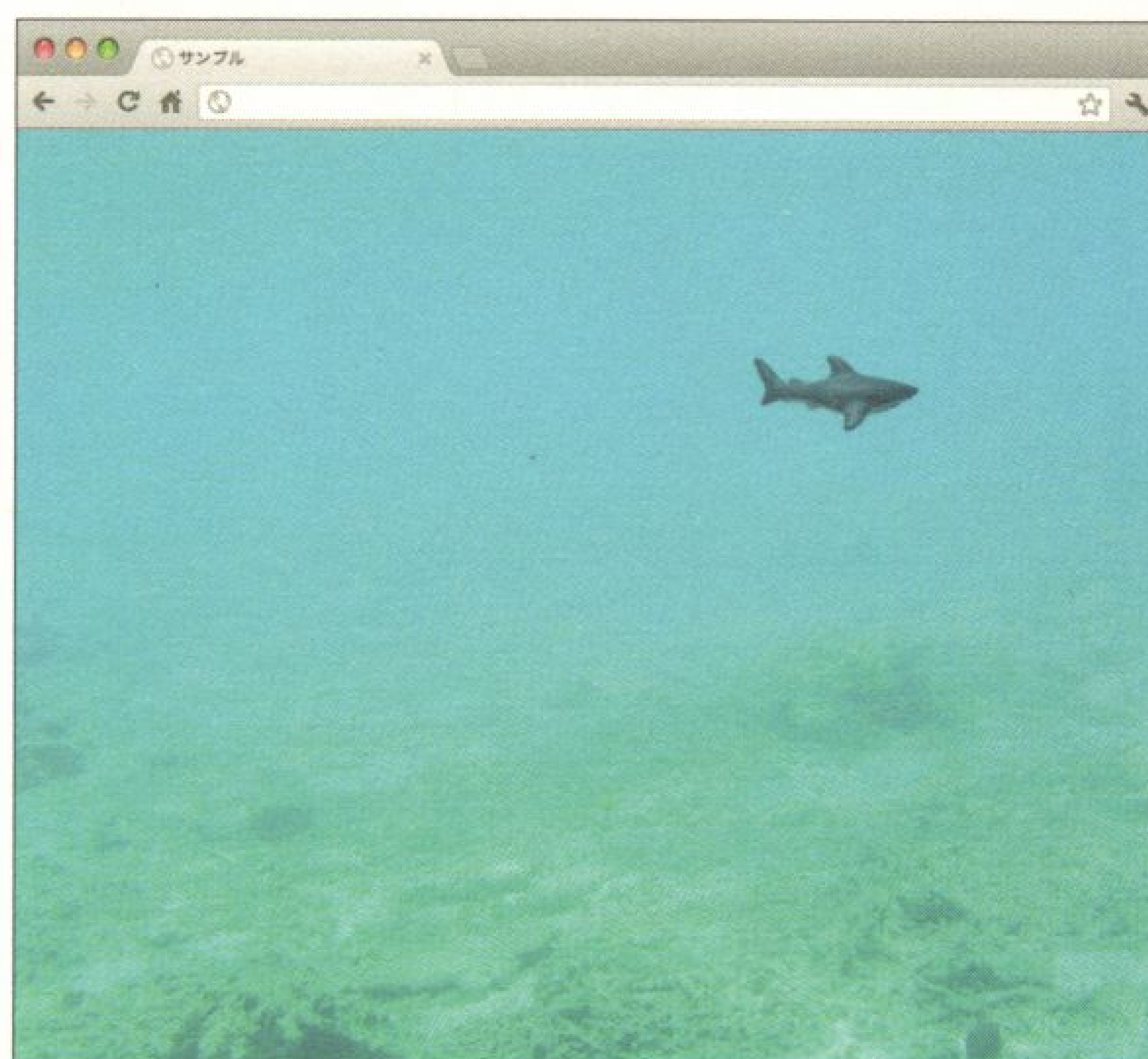
CSS　sample/chapter-11/lecture-11-3/03-animation.css

```
html, body { height: 100%; }
body {
  margin: 0;
  background: url(sea.jpg);
  -webkit-background-size: cover;
  -moz-background-size: cover;
  background-size: cover;
}
img {
  margin-top: 100px;
  width: 300px;
  height: 150px;
  -webkit-animation: shark 10s infinite;
}
@-webkit-keyframes shark {
  0% {
    -webkit-transform: scale(1.0);
  }
  20% {
    opacity: 0.6;
    -webkit-transform: translateX(500px) scale(0.2);
  }
  21% {
    opacity: 0.5;
    -webkit-transform: translateX(500px) rotate(-90deg) scale(0.2);
  }
  23% {
    opacity: 0.5;
    -webkit-transform: translate(500px, -500px) rotate(-90deg) scale(0.2);
  }
  24% {
    -webkit-transform: translate(-500px, -500px);
  }
  25% {
    -webkit-transform: translate(-500px);
  }
  98% {
    -webkit-transform: translate(-300px) scale(6.0);
  }
  100% {
    -webkit-transform: scale(1.0);
  }
}
```

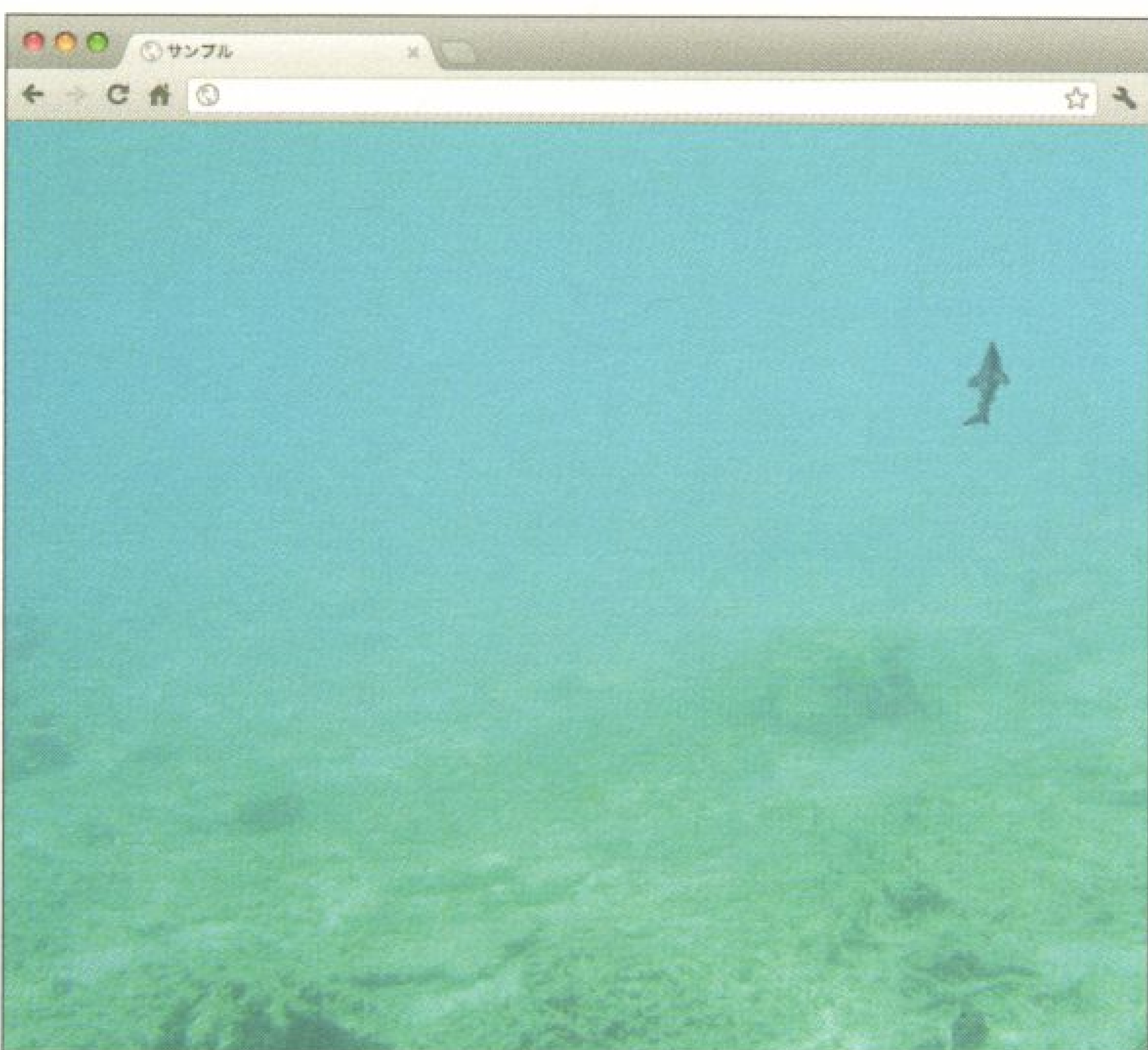
animationプロパティを使用したサンプル



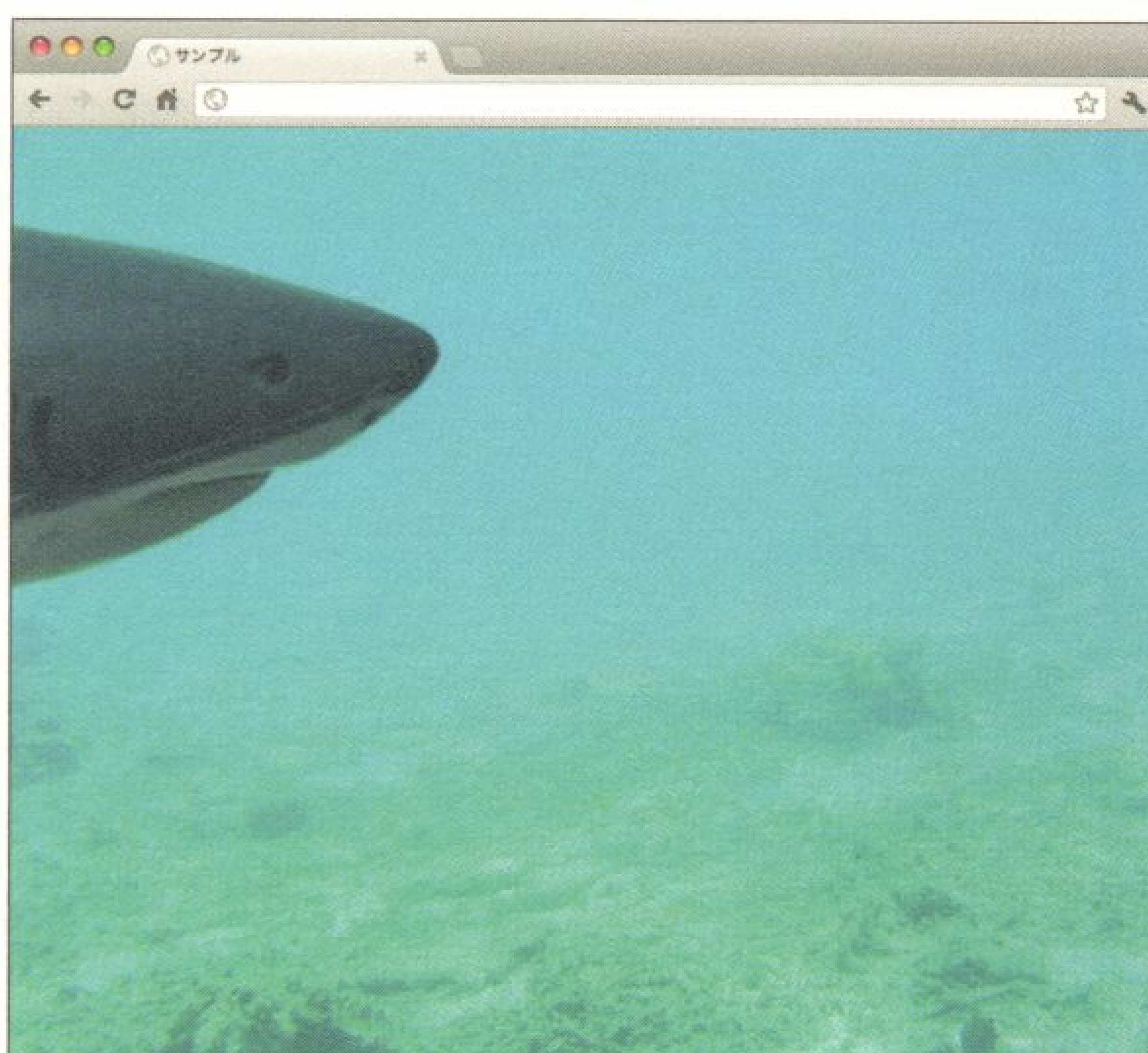
①サンプルをブラウザで開いたときの状態



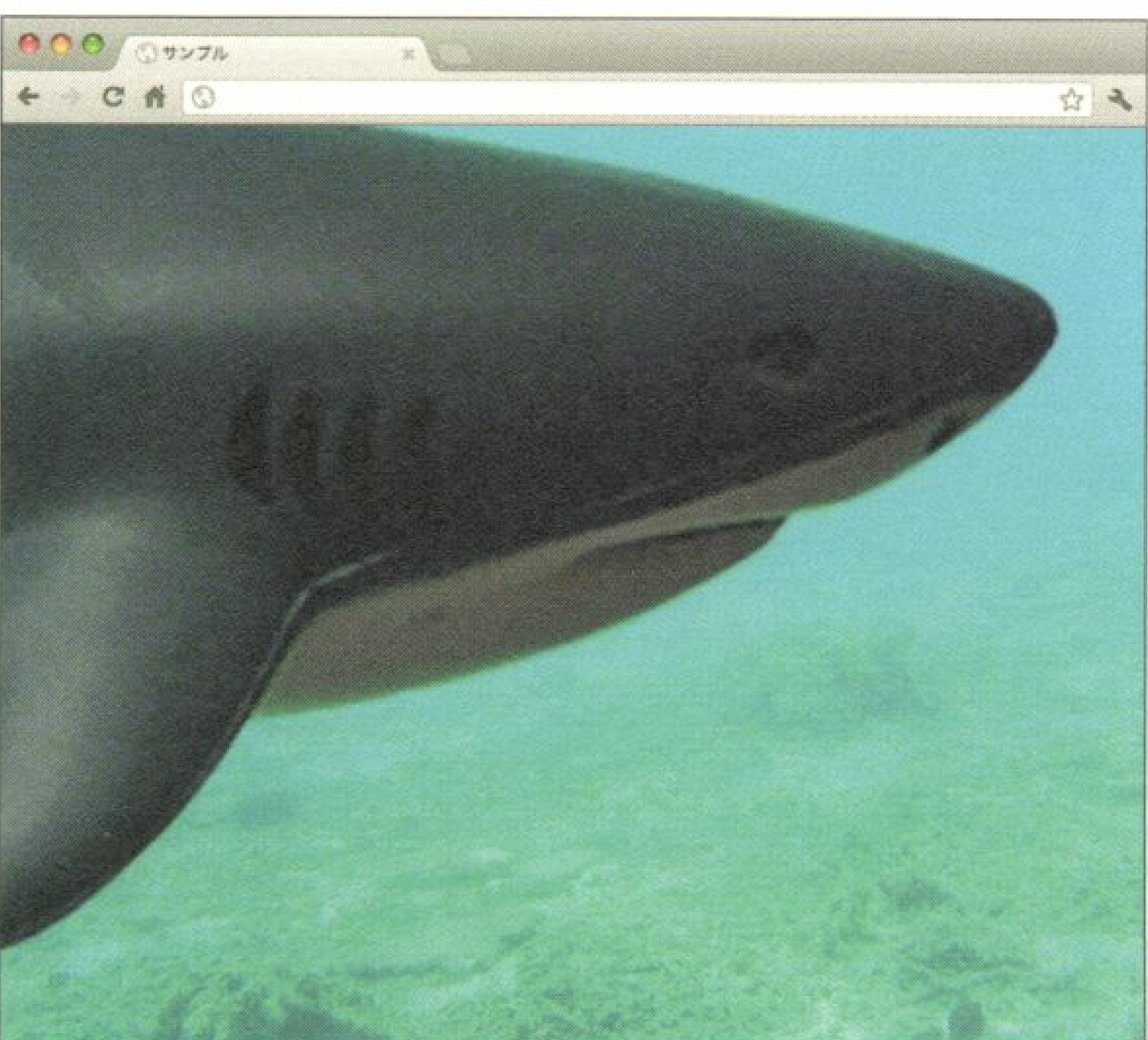
②サメはまず、半透明で小さくなりながら右側に移動する



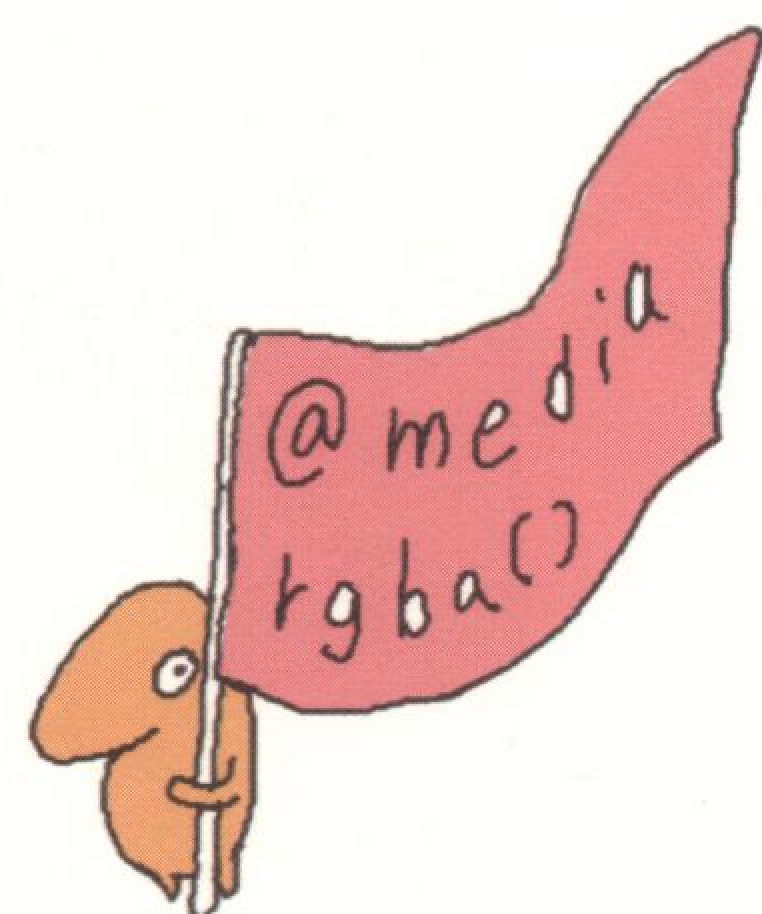
③右に500ピクセル進むと上を向き、画面上部へと一旦消える



④画面左側から徐々にあらわれる



⑤これでアニメーションの終了。これを何度も繰り返す


@media
rgba()

CHAPTER 12

ページをまるごと作ってみよう

ここまでは、HTML5とCSS3に含まれる機能を個別に説明してきました。Chapter 12では、それらを組み合せてページ全体を作っていく流れを見てみましょう。実際の制作においては、全体から細部にわたる設計のほか、各種コンテンツの準備なども必要となりますが、ここではそれらがすでに用意されているものとして進めていきます。

PRACTICE 12-1

サンプルページの概要を理解する

難易度：★☆☆☆☆

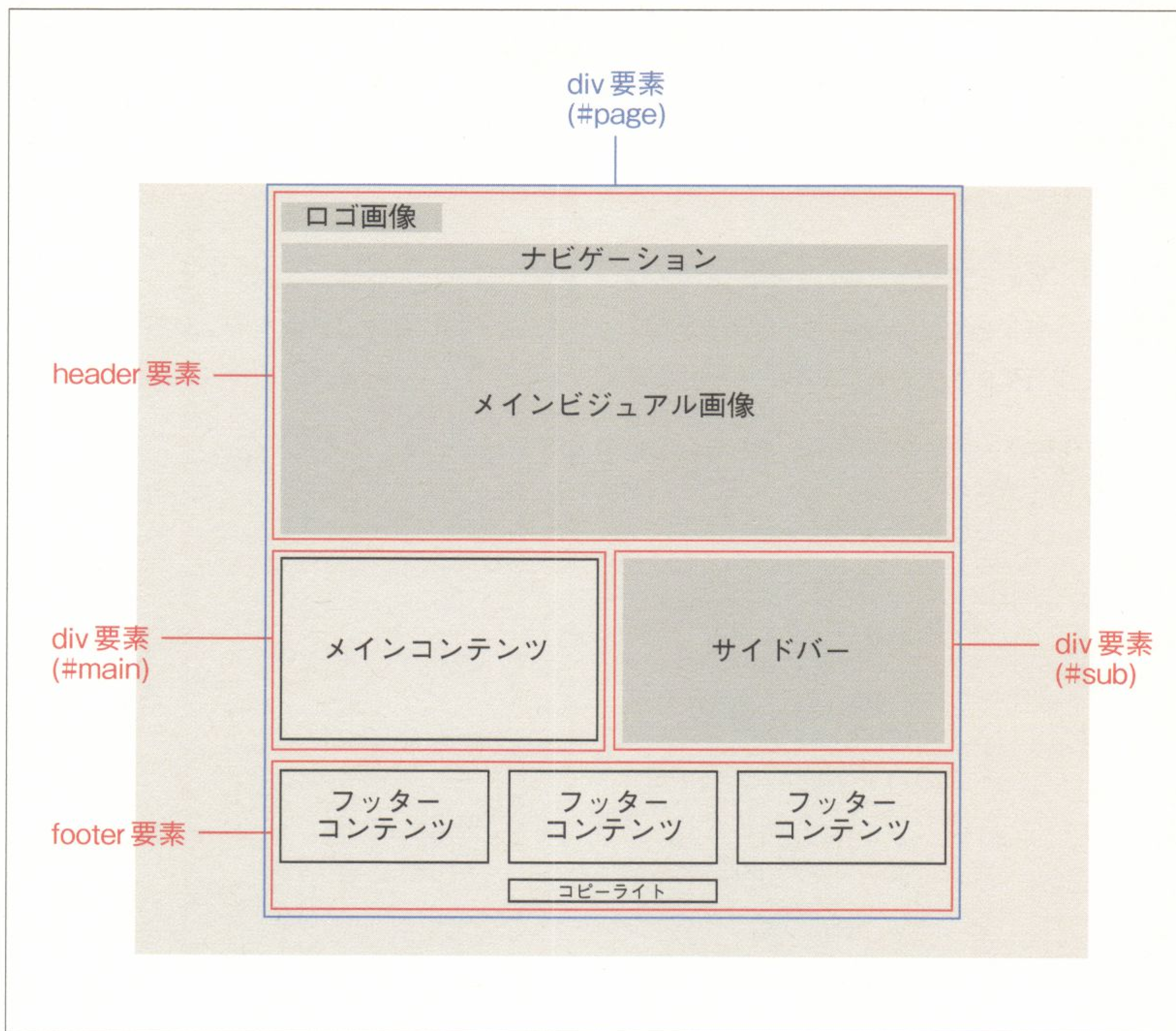
まずは、この章で制作するサンプルページについて紹介します。ここまで学習してきたことをどのように組み合わせて使うのか、流れに沿って理解できるようにしてみましょう。

1 この章で制作するサイトは、次のようなものです。意図的に、最近の日本のサイトでは比較的多いパターンのレイアウトにしています。あくまでもサンプルとして制作するページですので、シンプルで汎用的なものに仕上げてあります。



これから作成するページ

2 このサンプルページの構造をおおまかに図にすると、次のようになっています。
テキストや画像などの素材はすでに用意してありますので、さっそくHTMLから作り込んでいきましょう。



ページ全体の概略構造

PRACTICE 12-2

まずはマークアップから

難易度：★★☆☆☆

最初に、テキストと画像を入れた基本の状態のHTMLのマークアップを整えましょう。レイアウトのためのタグ付けも行います。

1 はじめに、素材（テキストと画像）を入れた状態のHTMLを確認します。と言っても、テキストには見出し（h1・h2要素）と段落（p要素）、リスト（ul要素）といった基本的な意味をあらわすタグはつけてあります。また、画像はimg要素として組み込んであります。

HTML　sample/chapter-12/practice-12-2/01.html

```
<!DOCTYPE html>
<html lang="ja">
<head>
<meta charset="utf-8">
<title>株式会社サンプル・サイト</title>
</head>
<body>

<img src="images/logo.png" alt="株式会社サンプル・サイト"
width="220" height="44">
<ul>
  <li><a href="#">ホーム</a></li>
  <li><a href="#">お知らせ</a></li>
  <li><a href="#">製品情報</a></li>
  <li><a href="#">サービス</a></li>
  <li><a href="#">会社概要</a></li>
  <li><a href="#">お問い合わせ</a></li>
</ul>
<img src="images/mainvisual.jpg" alt="日本のサンプルのリーディ
ング・カンパニーを目指します。" width="900" height="350">

<h1>サンプルだからこその“カタチ”がある</h1>
<p>わかりやすいサンプルはどうあるべきなのか？　その<a href="#">答え
</a>を知っているかどうかで<a href="#">サンプルの出来映え</a>は決
まってきます。これはサンプルのテキストです。…中略…　これはサンプルのテ
キストです。
</p>
<p>これはサンプルのテキストです。…中略…　これはサンプルのテキストです。
</p>

<h2>最高のサンプルを驚きのプライスで！</h2>
<p><img src="images/yen.png" width="45" height="45"
alt="">
これはサンプルのサイドバーのテキストです。…中略…　テキストです。
</p>
```

ロゴ画像（09行）
ナビゲーション（10〜17行）
メインビジュアル画像（18行）
メインコンテンツ（20〜24行）
サイドバー（26行〜）

```
<h2>CM「サンプルプルルン♪」メイキング公開</h2>
<p><img src="images/play.png" width="45" height="45"
alt="">
これはサンプルのサイドバーのテキストです。…中略… テキストです。
</p>

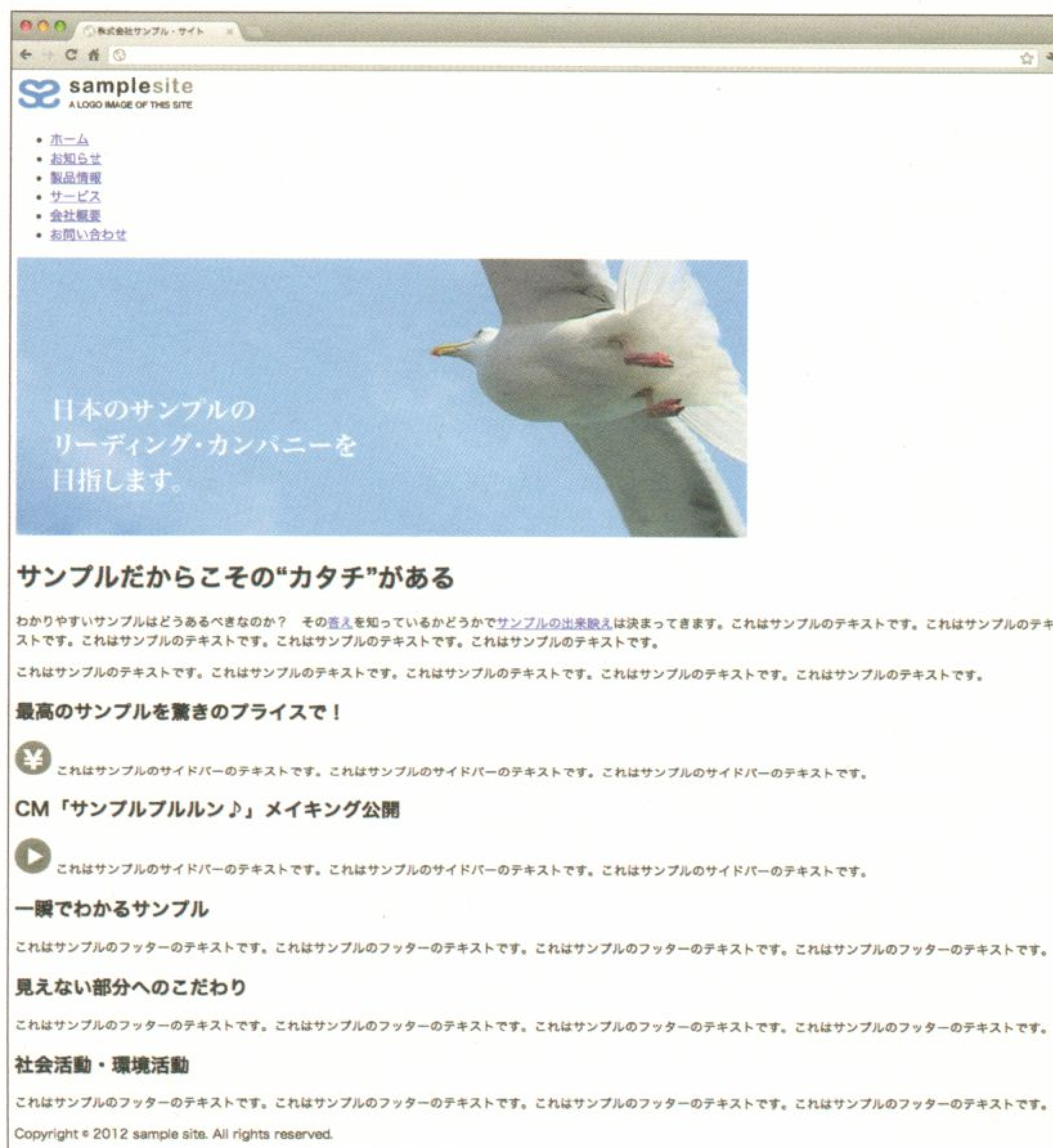
<h2>一瞬でわかるサンプル</h2>
<p>
これはサンプルのフッターのテキストです。…中略… テキストです。
</p>
<h2>見えない部分へのこだわり</h2>
<p>
これはサンプルのフッターのテキストです。…中略… テキストです。
</p>
<h2>社会活動・環境活動</h2>
<p>
これはサンプルのフッターのテキストです。…中略… テキストです。
</p>
<p>
Copyright &copy; 2012 sample site. All rights reserved.
</p>

</body>
</html>
```

フッターコンテンツ

コピーライト

HTMLは素材のテキストと画像を組み込んだだけの状態になっている



参考までに、この段階でブラウザで表示させるとこのようになっている

ワンポイント

このサンプルではリンク先のページは用意してありませんので、リンク先のURL（a要素のhref属性の値）はすべて「#」を入れただけの状態にしてあります。

■ h1・h2要素 ………… p.072へ
■ p要素 ………………… p.073へ
■ ul要素 ………………… p.191へ
■ img要素 ……………… p.130へ

2 次に、さきほど「ページ全体の概略構造」という図で示したレイアウトにするためのタグをつけます。各部分をまとめて配置できるようにするために、div要素やheader要素、footer要素でグループ化します。

HTML　sample/chapter-12/practice-12-2/02.html

```
<!DOCTYPE html>
<html lang="ja">
<head>
<meta charset="utf-8">
<title>株式会社サンプル・サイト</title>
</head>
<body>

<div id="page">

<header>
<img src="images/logo.png" alt="株式会社サンプル・サイト" width="220" height="44">
<ul>
  <li><a href="#">ホーム</a></li>
  <li><a href="#">お知らせ</a></li>
  <li><a href="#">製品情報</a></li>
  <li><a href="#">サービス</a></li>
  <li><a href="#">会社概要</a></li>
  <li><a href="#">お問い合わせ</a></li>
</ul>
<img src="images/mainvisual.jpg" alt="日本のサンプルのリーディング・カンパニーを目指します。" width="900" height="350">
</header>

<div id="main">
<h1>サンプルだからこその“カタチ”がある</h1>
<p>わかりやすいサンプルはどうあるべきなのか？　その<a href="#">答え</a>を知っているかどうかで<a href="#">サンプルの出来映え</a>は決まってきます。これはサンプルのテキストです。…中略…　これはサンプルのテキストです。
</p>
<p>これはサンプルのテキストです。…中略…　これはサンプルのテキストです。
</p>
</div>

<div id="sub">
<h2>最高のサンプルを驚きのプライスで！</h2>
<p><img src="images/yen.png" width="45" height="45" alt="">
これはサンプルのサイドバーのテキストです。…中略…　テキストです。
</p>
<h2>CM「サンプルプルルン♪」メイキング公開</h2>
<p><img src="images/play.png" width="45" height="45" alt="">
これはサンプルのサイドバーのテキストです。…中略…　テキストです。
</p>
</div>

<footer>
<h2>一瞬でわかるサンプル</h2>
```

```
<p>
これはサンプルのフッターのテキストです。…中略… テキストです。
</p>
<h2>見えない部分へのこだわり</h2>
<p>
これはサンプルのフッターのテキストです。…中略… テキストです。
</p>
<h2>社会活動・環境活動</h2>
<p>
これはサンプルのフッターのテキストです。…中略… テキストです。
</p>
<p>
Copyright &copy; 2012 sample site. All rights reserved.
</p>
</footer>
</div>
</body>
</html>
```

291ページの「ページ全体の概略構造」という図で示したレイアウトにするために必要な、グルーピングのためのタグ(赤と青で示したタグ)をつける

■ div要素 …………………………… p.074へ
■ header要素 ………………………… p.128へ
■ footer要素………………………… p.128へ

3 この段階ですでにHTMLは文法的にも問題ないものとなっていますが、せっかくですのでHTML5から新しく導入されたセクション関連のタグ(section要素・aside要素・nav要素)もつけてみましょう。HTML5で意味の変更されたsmall要素もコピーライトの部分で使ってみます。
さらに、これから指定していく「style.css」を読み込ませるためのlink要素を追加し、追加したタグに合わせてインデントを整えるとHTMLの完成です。最終的にHTMLのソースコードは、次のようになります。

HTML　　sample/chapter-12/practice-12-2/03.html

```
<!DOCTYPE html>
<html lang="ja">
<head>
<meta charset="utf-8">
<title>株式会社サンプル・サイト</title>
<link rel="stylesheet" href="styles.css">
</head>
<body>
<div id="page">
```

```
<header>
  <img src="images/logo.png" alt="株式会社サンプル・サイト" width="220" height="44">
  <nav>
    <ul>
      <li><a href="#">ホーム</a></li>
      <li><a href="#">お知らせ</a></li>
      <li><a href="#">製品情報</a></li>
      <li><a href="#">サービス</a></li>
      <li><a href="#">会社概要</a></li>
      <li><a href="#">お問い合わせ</a></li>
    </ul>
  </nav>
  <img src="images/mainvisual.jpg" alt="日本のサンプルのリーディング・カンパニーを目指します。" width="900" height="350">
</header>

<div id="main">
  <section>
    <h1>サンプルだからこその“カタチ”がある</h1>
    <p>わかりやすいサンプルはどうあるべきなのか？　その<a href="#">答え</a>を知っているかどうかで<a href="#">サンプルの出来映え</a>は決まってきます。これはサンプルのテキストです。…中略…　これはサンプルのテキストです。
    </p>
    <p>これはサンプルのテキストです。…中略…　これはサンプルのテキストです。
    </p>
  </section>
</div>

<div id="sub">
  <aside>
    <section>
      <h2>最高のサンプルを驚きのプライスで！</h2>
      <p><img src="images/yen.png" width="45" height="45" alt="">
これはサンプルのサイドバーのテキストです。…中略…　テキストです。
      </p>
    </section>
    <section>
      <h2>CM「サンプルプルルン♪」メイキング公開</h2>
      <p><img src="images/play.png" width="45" height="45" alt="">
これはサンプルのサイドバーのテキストです。…中略…　テキストです。
      </p>
    </section>
  </aside>
</div>

<footer>
  <aside>
    <section>
      <h2>一瞬でわかるサンプル</h2>
      <p>
これはサンプルのフッターのテキストです。…中略…　テキストです。
      </p>
```

```
    </section>
    <section>
      <h2>見えない部分へのこだわり</h2>
      <p>
これはサンプルのフッターのテキストです。…中略… テキストです。
      </p>
    </section>
    <section>
      <h2>社会活動・環境活動</h2>
      <p>
これはサンプルのフッターのテキストです。…中略… テキストです。
      </p>
    </section>
  </aside>
  <p>
    <small>Copyright &copy; 2012 sample site. All rights reserved.</small>
  </p>
</footer>

</div>

</body>
</html>
```

完成したHTMLのソースコード

■ section要素 ………………………… p.127へ
■ aside要素 ………………………… p.127へ
■ nav要素 ………………………… p.190へ

PRACTICE 12-3

ページ全体の初期設定と枠組みをつくる

難易度：★★★☆☆

続いて、CSSでページの見た目を整えていきましょう。文字や色などの基本の設定を読み込んだあと、2段組み、3段組みなどページ全体の枠組みを整えるための設定を行います。

1 さて、ここからCSSを指定していきます。それに先だって、ページ全体の初期設定となる、ごく基本的な指定を先に組み込んでおきましょう。簡単に言えば、ページ全体（body要素）の余白、フォントサイズ、文字色と背景色、行間の設定などの指定です。

どれも説明の必要がないくらい基本的なものばかりですが、HTML5から追加されたセクション関連要素に「display: block;」を指定している部分は重要です。実は、HTML5の新要素に未対応のブラウザでは、この指定がないとセクション関連要素はインライン要素のように表示されてしまうのです（displayプロパティの初期値は「inline」であるため）。古いブラウザで表示がおかしくなってしまうことを避けるために、この指定は忘れずに書き込んでおいてください。

CSS sample/chapter-12/practice-12-3/01-styles.css

```
@charset "utf-8";

/* 初期設定
---------------------------------------------------- */
body {
  margin: 0;
  padding: 0;
  font-size: small;
  color: #333;
  background: #fff;
}
h1, h2 {
  color: #222;
}
h1 {
  font-size: x-large;
}
h2 {
  font-size: medium;
}
p {
  line-height: 1.7;
```

```
}
a:link, a:visited {
  color: #37c;
}
a:hover {
  color: #999;
}
section, article, aside, nav, header, footer, hgroup {
  display: block;
}

/* ページの枠組み
---------------------------------------------------- */

/* ヘッダ
---------------------------------------------------- */

/* ナビゲーション
---------------------------------------------------- */

/* サイドバー
---------------------------------------------------- */

/* フッタ
---------------------------------------------------- */
```

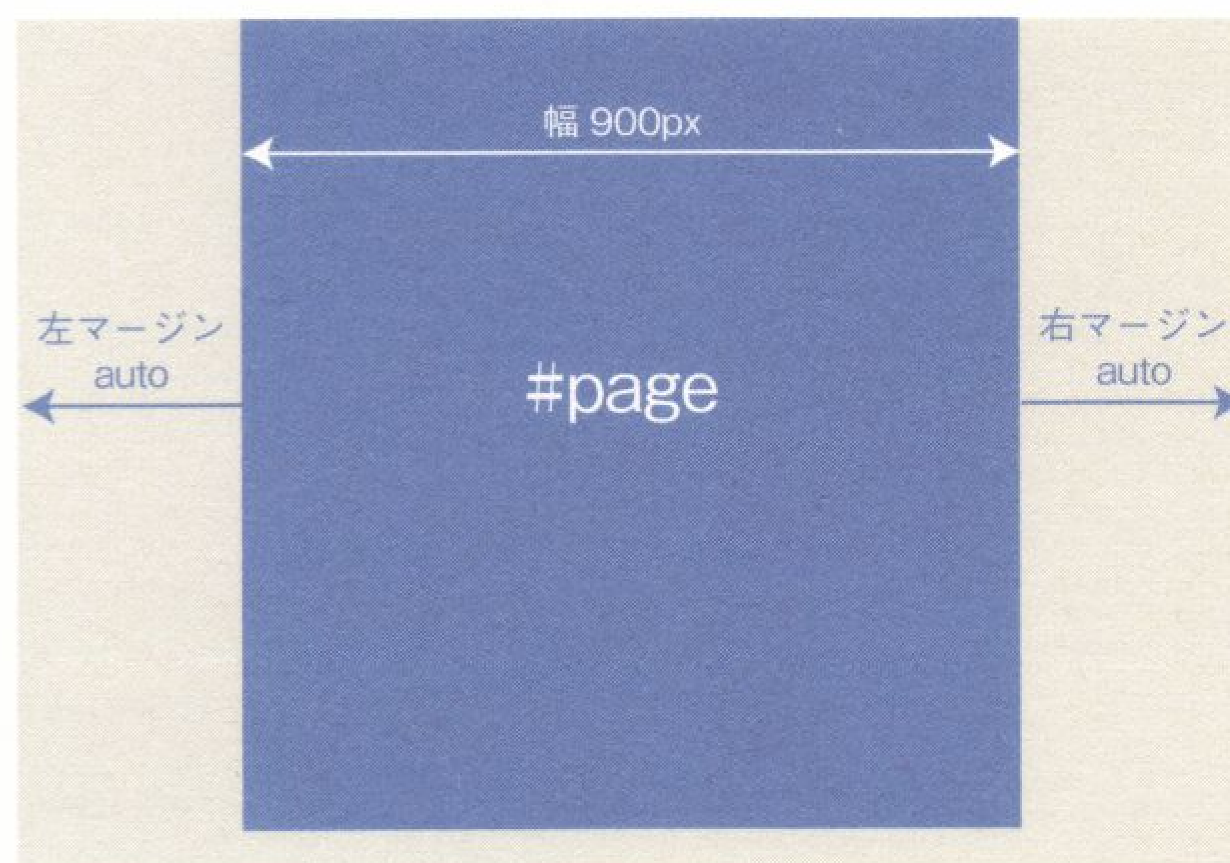
セクション関連要素に対する指定

この段階での「styles.css」のソースコード。初期設定となるソースコードのほかに、これ以降に入力するソースコードの見出しとなるコメントも入れてある

---

2 続けて、ページ全体の枠組みを設定します。ここでまずコンテンツ全体の幅を決めて、それを中央寄せにして配置し、メインコンテンツとサイドバーを横に並べ、フッターの内部にある３段組みも済ませます。それらの枠組みの中身については、このあとに順番に指定していきます。

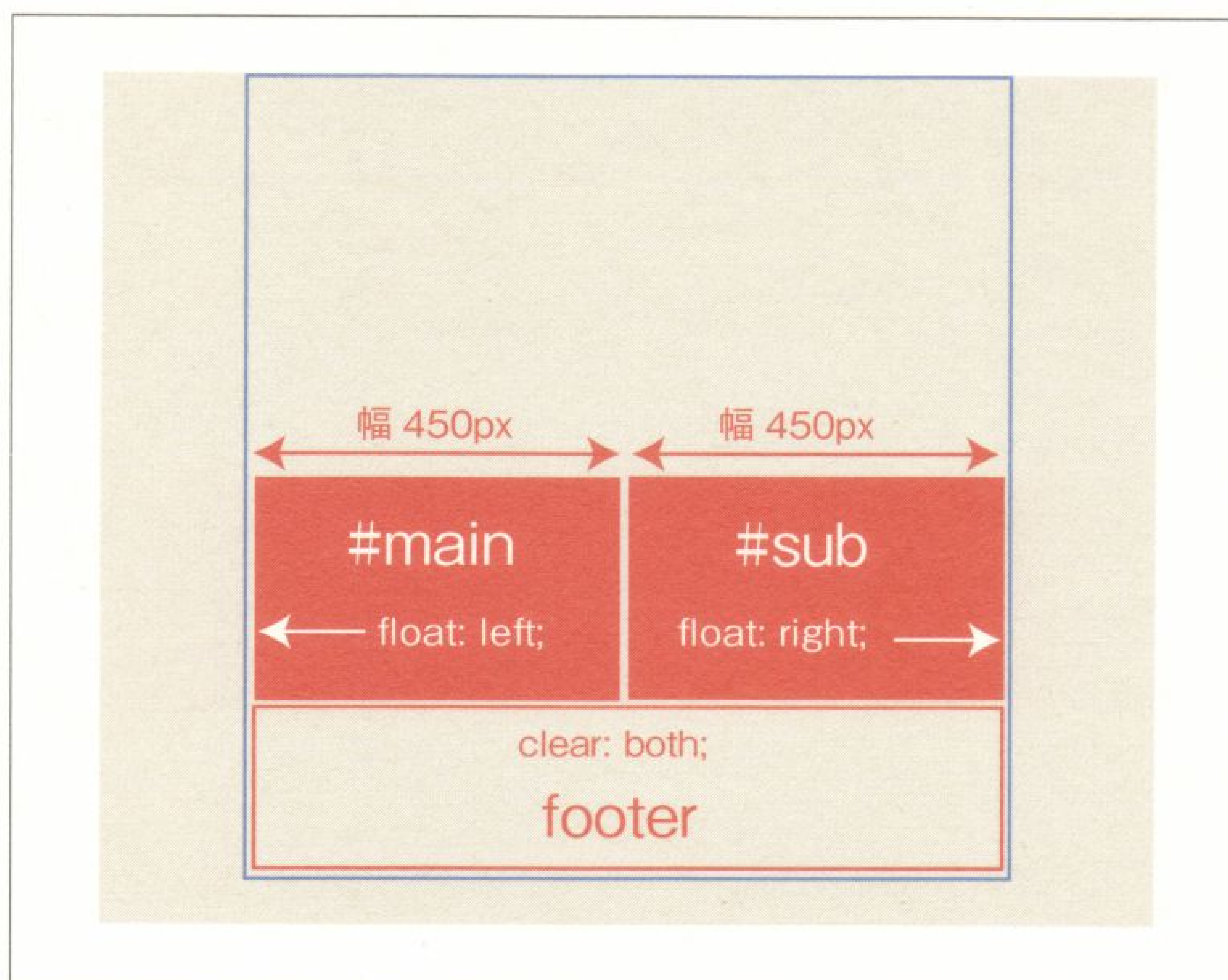
まず、body要素内のすべての要素が入っている#pageの幅を900ピクセルにします。これでコンテンツ全体の幅が900ピクセルに制限されます。#pageの左右のマージンを「auto」にすると、左右のマージンは同じ距離となるのでセンタリングされます。



コンテンツ全体の幅は900ピクセルに設定。左右のマージンを「auto」にして中央寄せで表示させる

CHAPTER 12

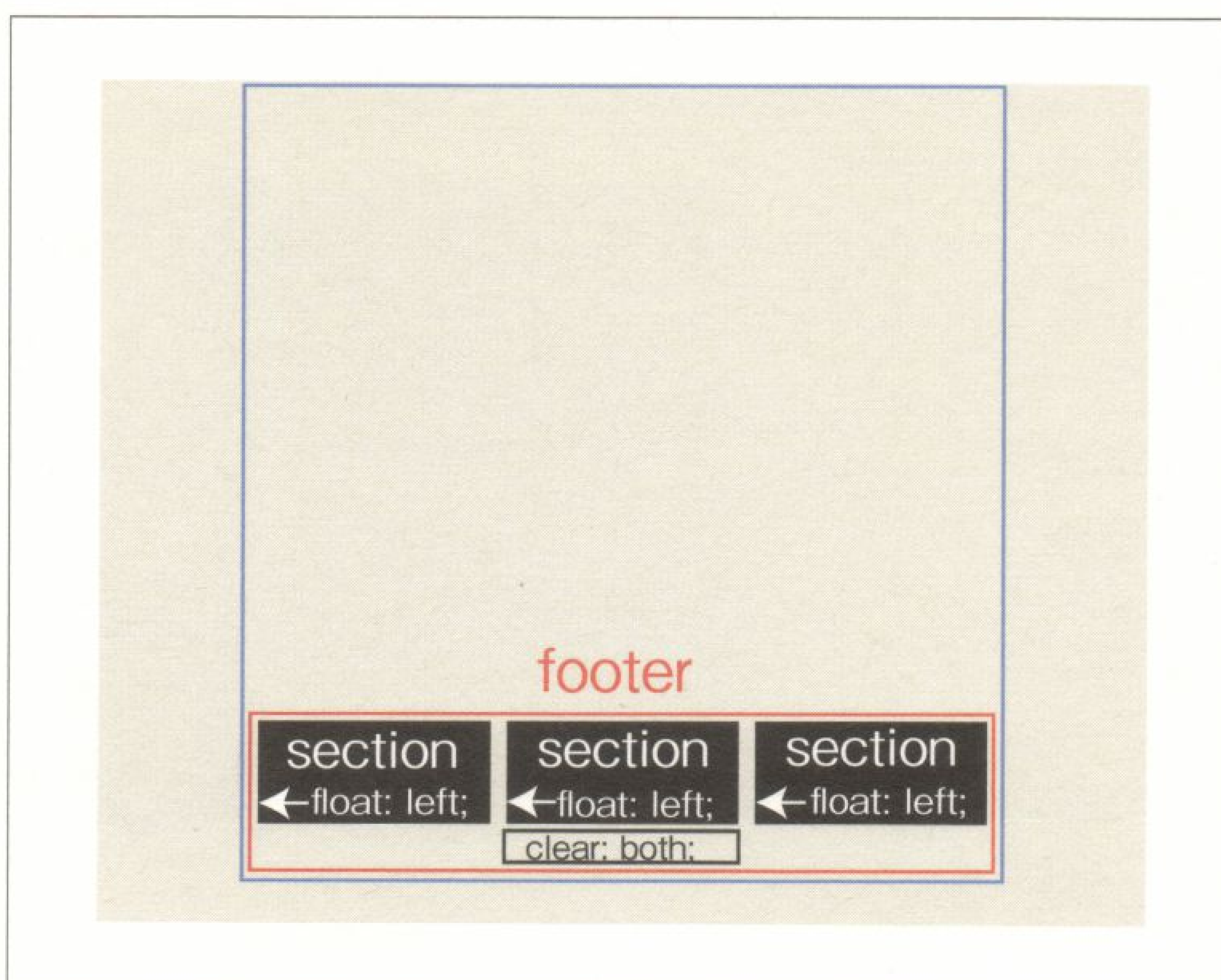
3 次に、メインコンテンツ(#main)とサイドバー(#sub)の幅をそれぞれ450ピクセル(900ピクセルの半分)にします。そして、#mainには「float: left;」、#subには「float: right;」を指定します。これで#mainと#subは横に並びます。これらの下にあるfooterでこれらのフロートはクリアしておきます。



#main#subの幅を450ピクセルに設定し、floatで左右に振り分けて横に並べる

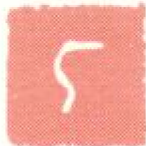
■ floatプロパティ ……… p.164へ
■ 2段組み ………………… p.168へ

4 フッター内のsection要素については、それぞれの幅を280ピクセルに設定し、すべてに「float: left;」を指定して横に並べます。フロートは、その下にあるコピーライトのテキストでクリアしておきます。



フッター内のsectionもフロートで横に並べ、その下のコピーライトのテキストでクリアする

■ 3段組み ………………… p.172へ



これらを指定したCSSのソースコードは、次の通りです。

CSS　sample/chapter-12/practice-12-3/02-styles.css（HTMLファイルは02.html）

```
・・・

/* ページの枠組み
---------------------------------------------------- */
#page {
  margin: 0 auto;
  width: 900px;
}
#main {
  float: left;
  width: 450px;
}
#sub {
  float: right;
  width: 450px;
}
footer {
  clear: both;
}
footer section {
  float: left;
  width: 280px;
}
footer section:nth-child(2) {
  margin: 0 30px;
}
#copyright {
  clear: both;
}

/* ヘッダー
---------------------------------------------------- */

・・・
```

フッター内の真ん中のセクションにだけ左右に30pxのマージンを指定

ページ全体の枠組みを指定しているCSSのソースコード（追加した部分だけ抜粋）

■ :nth-child() ……………………… p.117へ

6 フッター内のセクションについては、3つの段が280ピクセルだと、合計は840ピクセルで、幅に60ピクセルの余裕があります。その幅は各セクション間の余白として、真ん中のセクションの左右に30ピクセルずつのマージンとして指定します(セレクタにnth-child(2)を使っている部分)。

この指定は、セレクタにnth-child()を使っているため、Internet Explorer 8以前には適用されません。それらにも対応させるには、nth-child()を使用せずに、真ん中のセクションだけにid属性またはclass属性を指定し、それをセレクタで指定する必要があります。具体的な方法については、このあとの「Practice 12-6　IE6対応へのヒント(p.314)」を参照してください。

この段階でブラウザで表示させると、次のようになっています。まだページ全体の初期設定と大枠の指定しかしていないのでおかしな部分は多くありますが、それなりの雰囲気は出てきました。



この時点でのソースコードをブラウザで表示させたところ

PRACTICE 12-4

ナビゲーションとヘッダー

難易度：★★★★☆

ナビゲーションを整えて行きます。今は箇条書きの状態になっているナビゲーションの項目を、横に並べ、メニューらしい見た目に整えます。また、ロゴ画像の表示も整えてヘッダー領域を完成させます。

1 次はヘッダーの中にあるナビゲーションを作成します。ナビゲーションは、一般的なページの中では制作にもっとも手間のかかる部分でもあります。ただし、この部分が出来上がると、Webページらしさがぐっと増してきます。

まずはナビゲーションの現在の状態を確認しておきましょう。まだこの部分にはCSSを一切していません。HTMLは、ul要素の各項目がリンクになっているだけのシンプルなものです。

HTML

```
・・・
<nav>
  <ul>
    <li><a href="#">ホーム</a></li>
    <li><a href="#">お知らせ</a></li>
    <li><a href="#">製品情報</a></li>
    <li><a href="#">サービス</a></li>
    <li><a href="#">会社概要</a></li>
    <li><a href="#">お問い合わせ</a></li>
  </ul>
</nav>
・・・
```

ナビゲーション部分のHTMLのソースコード



CSSを指定する前の段階のナビゲーション

displayプロパティ ………… p.203へ

2 はじめに、一般的なリストの表示形式を、ブロックレベル要素が横に並んでいる状態にします。次の赤で示した指定を追加してください。li要素とa要素の両方に「display: block;」を指定していますが、これによってli要素はブロックレベルの表示になって行頭記号が消え、a要素は幅と高さが指定できるようになります(インライン要素のテキストには幅も高さも指定できません)。

CSS

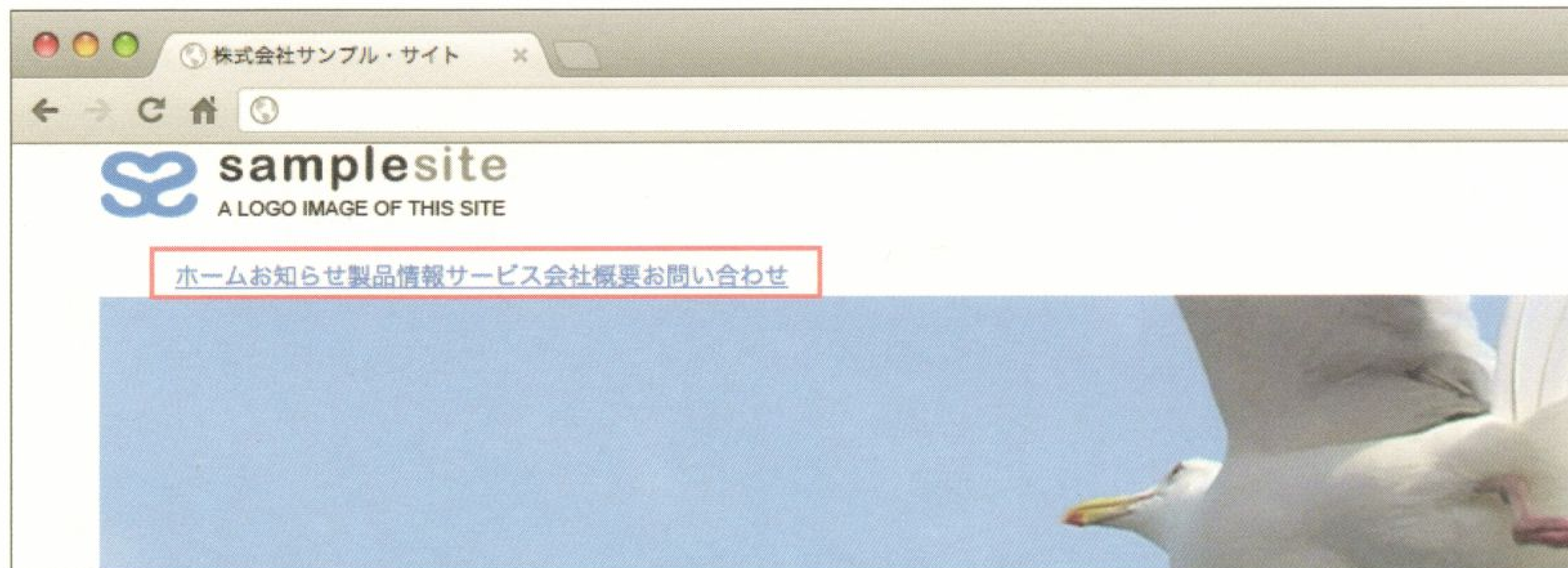
```
・・・

/* ナビゲーション
------------------------------------------------------ */
nav li, nav a {
  display: block;
}
nav li {
  float: left;
}

/* サイドバー
------------------------------------------------------ */

・・・
```

リストの内容をすべてブロックにして、floatで横に並べる



この段階でのナビゲーションの表示

3 続けて、横に並んだ各項目がナビゲーションらしく見えるように形を整えます。ここで追加するのは、次のソースコードの赤で示した部分です。最初にul要素に「overflow: hidden;」を指定しているのは、高さが0の状態から、フロートしているすべてのli要素を含むように高さを拡張するためです(ここではclearfixを使用せずにoverflowプロパティを使用しているということです)。そして、マージンとパディングを調整して、ナビゲーションとその下の画像との間隔を10ピクセルにしています。さらに、全体をグレーのボーダーで囲い、行間をフォントサイズと同じにして、

薄いグレーのグラデーションの背景画像を表示させています[※16]。

li要素には、右側にグレーのボーダーを指定して、各項目の区切る線として表示させます。そして、a要素にパディングを指定して余白を広くすると、ナビゲーションらしい表示に変わります。

CSS

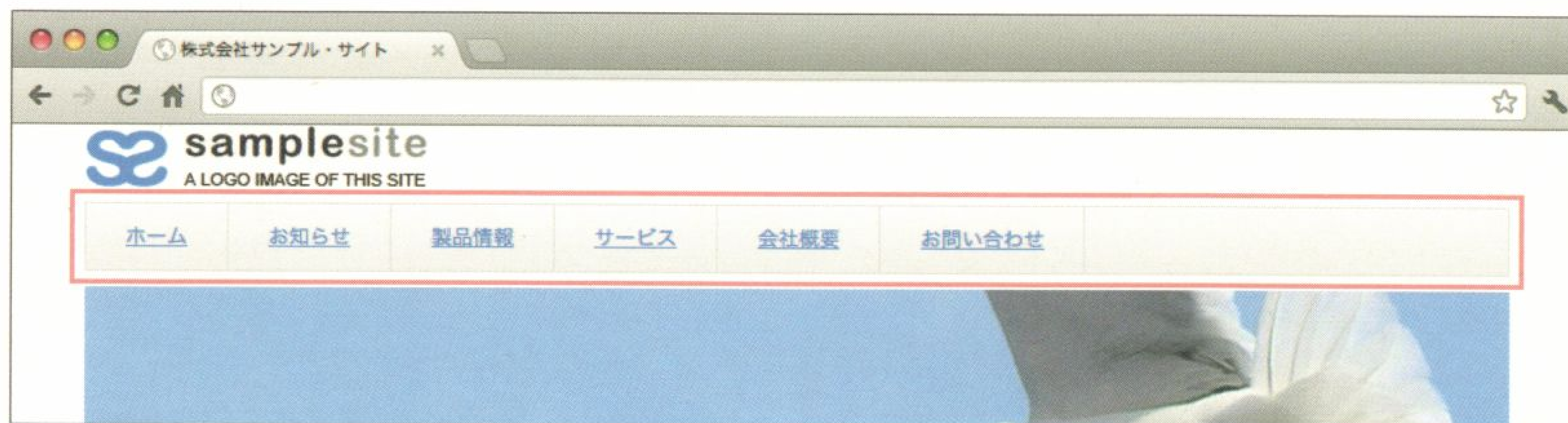
```
・・・

/* ナビゲーション
--------------------------------------------------- */
nav ul {
  overflow: hidden;
  margin: 0 0 10px 0;
  padding: 0;
  border: 1px solid #ddd;
  line-height: 1.0;
  background: url(images/li-bg.jpg) repeat-x bottom;
}
nav li, nav a {
  display: block;
}
nav li {
  float: left;
  border-right: 1px solid #ddd;
}
nav a {
  padding: 15px 25px;
}

/* サイドバー
--------------------------------------------------- */

・・・
```

余白、背景、ボーダーなどを指定してナビゲーションらしくする



この段階でのナビゲーションの表示

overflowプロパティ ………………… p.207へ

※16：グラデーションはCSS3にも含まれており、画像を使わなくてもCSSで指定することが可能です。ただし、グラデーションは大きな仕様変更が何度もおこなわれており、その安定性や互換性の面からあえて画像を使用しています。

4 さらに細かい指定を追加して、ナビゲーション部分を完成させます。まず、ナビゲーションの最初の項目(現在表示しているページの項目)の表示を変えるために、現在表示しているページのli要素には「id="current"」をつけることにします(次のCSSのソースコードの最後で文字色と背景色を指定しています)。

ul要素にはborder-radiusプロパティを指定して角を丸くします。そして、左上の内側に白い影、右下の外側に半透明の黒い影を表示させます。a要素の文字を太字にして下線を消し、右下に白い影を表示させて文字がへこんでいるような効果を出します。:hoverの指定を加えて、カーソルがのったときに背景が少し薄くなるように別の背景画像を指定するとナビゲーションの完成です。

HTML

```
・・・
<nav>
  <ul>
    <li id="current"><a href="#">ホーム</a></li>
    <li><a href="#">お知らせ</a></li>
    <li><a href="#">製品情報</a></li>
    <li><a href="#">サービス</a></li>
    <li><a href="#">会社概要</a></li>
    <li><a href="#">お問い合わせ</a></li>
  </ul>
</nav>
・・・
```

CSS

```
・・・

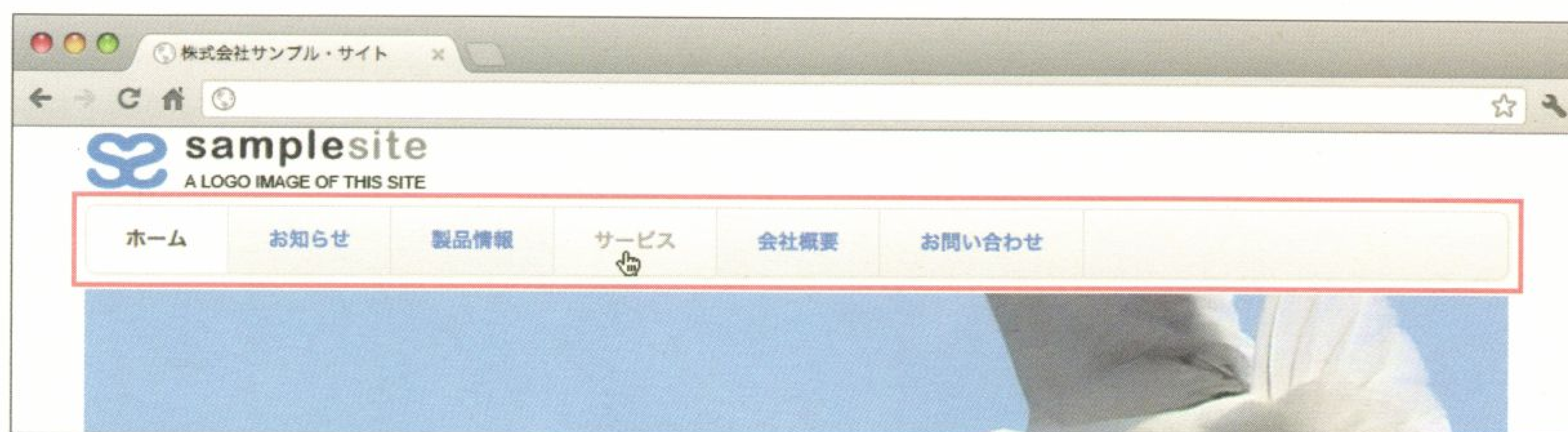
/* ナビゲーション
---------------------------------------------------- */
nav ul {
  overflow: hidden;
  margin: 0 0 10px 0;
  padding: 0;
  border: 1px solid #ddd;
  -webkit-border-radius: 7px;
  -moz-border-radius: 7px;
  border-radius: 7px;
  -webkit-box-shadow: 1px 1px 0 #fff inset, 1px 1px 3px rgba(0, 0, 0, 0.1);
  -moz-box-shadow: 1px 1px 0 #fff inset, 1px 1px 3px rgba(0, 0, 0, 0.1);
  box-shadow: inset 1px 1px 0 #fff, 1px 1px 3px rgba(0, 0, 0, 0.1);
  line-height: 1.0;
  background: url(images/li-bg.jpg) repeat-x bottom;
}
nav li, nav a {
  display: block;
}
nav li {
```

```
  float: left;
  border-right: 1px solid #ddd;
}
nav a {
  padding: 15px 25px;
  font-weight: bold;
  text-decoration: none;
  text-shadow: 1px 1px 0 #fff;
}
nav a:hover {
  background: url(images/li-bg-hover.jpg) repeat-x bottom;
}
nav #current a {
  color: #333;
  background: #fff;
}

/* サイドバー
--------------------------------------------------- */

...
```

ul要素を角丸にして影を表示させ、細かい部分の表示を整える



ナビゲーションが完成した

■ border-radius プロパティ ………… p.148へ
■ :hover ……………………………… p.113へ

5

ナビゲーションは完成したものの、その上にあるロゴ画像はちょっと窮屈な感じのままです。この部分の表示を整えてヘッダー領域を完成させましょう。

ヘッダー領域内にはロゴ画像とメインビジュアル画像の2つが入っていますが、それらは何も指定しなければインラインの状態で、画像の下に隙間ができるなど取り扱いが面倒です。そこで、ヘッダー領域内の画像に「display: block;」を指定して、両方の画像をブロックレベルに変更します。こうすることで画像の下の隙間も消えます。

本来、2つのimg要素があって、その1つめに対して何かを指定するには、:nth-of-type() などのセレクタを使えばいいのですが、それを使うとInternet Explorer 8以前には指定が効かないことになってしまいます。そこで、ここでは最初の画像に「id="logo"」を追加し、それに対して上下に20ピクセルずつのマージンを指定しています。これで、ヘッダー部分の指定がすべて完了しました。

HTML　sample/chapter-12/practice-12-4/index.html

```
・・・

<div id="page">

<header>
  <img id="logo" src="images/logo.png" alt="株式会社サンプル・サイト" width="220" height="44">
  <nav>
    <ul>
      <li id="current"><a href="#">ホーム</a></li>
      <li><a href="#">お知らせ</a></li>
・・・
```

CSS　sample/chapter-12/practice-12-4/styles.css

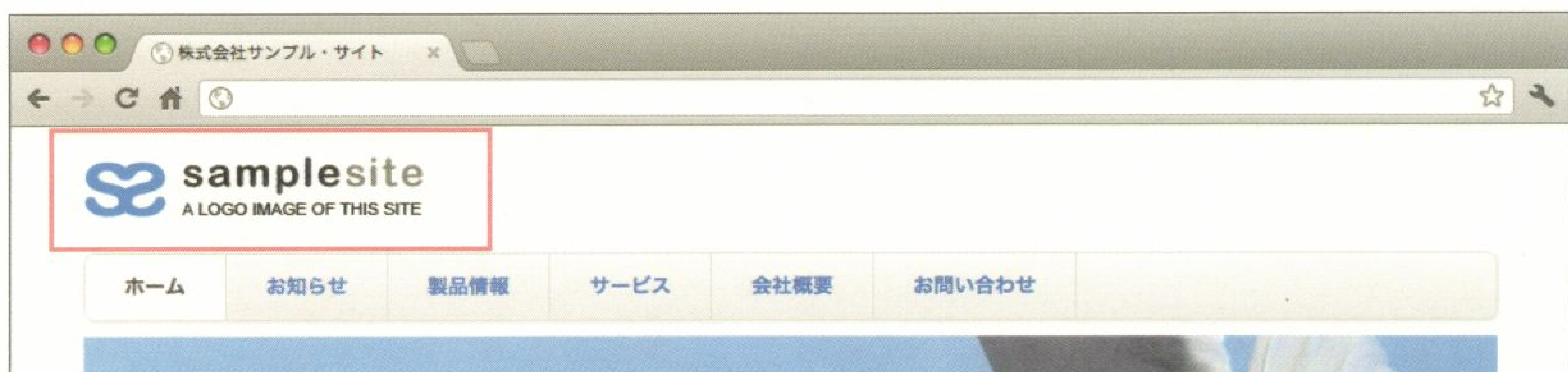
```
・・・

/* ヘッダー
------------------------------------------------------ */
header img {
  display: block;
}
#logo {
  margin: 20px 0;
}

/* ナビゲーション
------------------------------------------------------ */

・・・
```

この指定を追加するとヘッダー部分はすべて完成



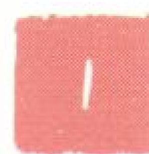
ロゴ画像の上下に余白が出来た

PRACTICE 12-5

サイドバーとフッター

難易度：★★★☆☆

最後にサイドバーとフッターエリアの表示を整えて完成です。サイドバーは背景色や余白、ボーダーなどを整え、アイコン画像とテキストの位置を調整します。フッターコンテンツは上部に横線を表示し、コピーライトの位置や色を調整します。



さて、あとはサイドバーとフッターの指定が完了すると、サンプルページは完成です。まずは現在のサイドバーとフッターの状態を確認しておきましょう。

HTML

```
・・・

<div id="sub">
  <aside>
    <section>
      <h2>最高のサンプルを驚きのプライスで！</h2>
      <p><img src="images/yen.png" width="45" height="45"
alt="">
これはサンプルのサイドバーのテキストです。これはサンプルのサイドバーの
テキストです。これはサンプルのサイドバーのテキストです。
      </p>
    </section>
    <section>
      <h2>CM「サンプルプルルン♪」メイキング公開</h2>
      <p><img src="images/play.png" width="45" height="45"
alt="">
これはサンプルのサイドバーのテキストです。これはサンプルのサイドバーの
テキストです。これはサンプルのサイドバーのテキストです。
      </p>
    </section>
  </aside>
</div>

<footer>
  <aside>
    <section>
      <h2>一瞬でわかるサンプル</h2>
      <p>
これはサンプルのフッターのテキストです。これはサンプルのフッターのテキ
ストです。これはサンプルのフッターのテキストです。これはサンプルのフッタ
ーのテキストです。
      </p>
    </section>
    <section>
```

サイドバー

CHAPTER 12

```
        <h2>見えない部分へのこだわり</h2>
        <p>
これはサンプルのフッターのテキストです。これはサンプルのフッターのテキストです。これはサンプルのフッターのテキストです。これはサンプルのフッターのテキストです。
        </p>
      </section>
      <section>
        <h2>社会活動・環境活動</h2>
        <p>
これはサンプルのフッターのテキストです。これはサンプルのフッターのテキストです。これはサンプルのフッターのテキストです。これはサンプルのフッターのテキストです。
        </p>
      </section>
    </aside>
    <p>
      <small>Copyright &copy; 2012 sample site. All rights reserved.</small>
    </p>
</footer>

・・・
```

フッター

サイドバーとフッター部分のHTMLのソースコード

リーディング・カンパニーを
目指します。

サンプルだからこその"カタチ"がある

わかりやすいサンプルはどうあるべきなのか？　その答えを知っているかどうかでサンプルの出来映えは決まってきます。これはサンプルのテキストです。これはサンプルのテキストです。これはサンプルのテキストです。これはサンプルのテキストです。これはサンプルのテキストです。

これはサンプルのテキストです。これはサンプルのテキストです。これはサンプルのテキストです。これはサンプルのテキストです。これはサンプルのテキストです。

最高のサンプルを驚きのプライスで！

これはサンプルのサイドバーのテキストです。これはサンプルのサイドバーのテキストです。これはサンプルのサイドバーのテキストです。

CM「サンプルプルルン♪」メイキング公開

これはサンプルのサイドバーのテキストです。これはサンプルのサイドバーのテキストです。これはサンプルのサイドバーのテキストです。

一瞬でわかるサンプル

これはサンプルのフッターのテキストです。これはサンプルのフッターのテキストです。これはサンプルのフッターのテキストです。これはサンプルのフッターのテキストです。

見えない部分へのこだわり

これはサンプルのフッターのテキストです。これはサンプルのフッターのテキストです。これはサンプルのフッターのテキストです。これはサンプルのフッターのテキストです。

社会活動・環境活動

これはサンプルのフッターのテキストです。これはサンプルのフッターのテキストです。これはサンプルのフッターのテキストです。これはサンプルのフッターのテキストです。

Copyright © 2012 sample site. All rights reserved.

この時点でのサイドバーとフッターの状態

2 サイドバーは、その全体の背景をグレーにするために、#sub内のaside要素に背景色を指定します。はじめに余白を調整してグレーのボーダーを表示させ、角丸にし、薄いグレーの背景を指定します。そして、ボックスの左上内側に白い影、右下外側に半透明の黒い影を表示させます。そしてtext-shadowプロパティで文字の右下に白い影を表示させて文字がへこんでいるような効果を出します（このあたりの指定はナビゲーション部分と同様です）。

サイドバーには2つの画像のアイコンがありますが、これらの横にテキストが回り込むように、画像には「float: left;」を指定します。念のため、サイドバー内の見出し（h2要素）でフロートをクリアしておき、見出しと画像の余白を調整するとサイドバーは完成です。

CSS

```
・・・

/* サイドバー
------------------------------------------------- */
#sub aside {
  margin: 20px 0 20px 20px;
  padding: 15px;
  border: 1px solid #ddd;
  -webkit-border-radius: 7px;
  -moz-border-radius: 7px;
  border-radius: 7px;
  background-color: #f5f5f5;
  -webkit-box-shadow: 1px 1px 0 #fff inset, 1px 1px 3px rgba(0, 0, 0, 0.1);
  -moz-box-shadow: 1px 1px 0 #fff inset, 1px 1px 3px rgba(0, 0, 0, 0.1);
  box-shadow: inset 1px 1px 0 #fff, 1px 1px 3px rgba(0, 0, 0, 0.1);
  text-shadow: 1px 1px 0 #fff;
}
#sub h2 {
  clear: both;
  margin-top: 0;
}
#sub img {
  float: left;
  margin: 0.4em 7px 0 0;
}

/* フッター
------------------------------------------------- */
```

サイドバーに指定するCSSのソースコード

■ box-shadowプロパティ ………… p.230へ
■ text-shadowプロパティ ………… p.094へ
■ clearプロパティ …………………… p.166へ

日本のサンプルの
リーディング・カンパニーを
目指します。

サンプルだからこその“カタチ”がある

わかりやすいサンプルはどうあるべきなのか？　その答えを知っているかどうかでサンプルの出来映えは決まってきます。これはサンプルのテキストです。これはサンプルのテキストです。これはサンプルのテキストです。これはサンプルのテキストです。これはサンプルのテキストです。

これはサンプルのテキストです。これはサンプルのテキストです。これはサンプルのテキストです。これはサンプルのテキストです。これはサンプルのテキストです。

最高のサンプルを驚きのプライスで！



これはサンプルのサイドバーのテキストです。これはサンプルのサイドバーのテキストです。これはサンプルのサイドバーのテキストです。

CM「サンプルプルルン♪」メイキング公開



これはサンプルのサイドバーのテキストです。これはサンプルのサイドバーのテキストです。これはサンプルのサイドバーのテキストです。

一瞬でわかるサンプル

これはサンプルのフッターのテキストです。これはサンプルのフッターのテキストです。これはサンプルのフッターのテキストです。これはサンプルのフッターのテキストです。

見えない部分へのこだわり

これはサンプルのフッターのテキストです。これはサンプルのフッターのテキストです。これはサンプルのフッターのテキストです。これはサンプルのフッターのテキストです。

社会活動・環境活動

これはサンプルのフッターのテキストです。これはサンプルのフッターのテキストです。これはサンプルのフッターのテキストです。これはサンプルのフッターのテキストです。

CSS指定後のサイドバーの表示

3 フッター領域の上には横線を表示させます。フッター領域全体を囲っているのはfooter要素ですので、footer要素の上のボーダーを表示させます。

次に、ページ一番下のコピーライト表記のテキストを調整します。コピーライト表記はp要素の中に入れられていますが、フッター内には全部で4つのp要素が入っています。CSS3では、そのうちの最後のp要素を指定するためのセレクタも用意されていますが、ここでは古いブラウザでも問題なく表示できるように、コピーライトの段落に「id="copyright"」を追加して、セレクタではそれを対象にすることにします。#copyrightの上の余白を調整し、中央揃えにして、文字色をグレーにするとサンプルページの完成です。

HTML　　sample/chapter-12/practice-12-5/index.html

```html
・・・
  <p id="copyright">
      <small>Copyright &copy; 2012 sample site. All rights reserved.</small>
  </p>
</footer>

・・・
```

CSS　sample/chapter-12/practice-12-5/styles.css

```
・・・

/* フッター
--------------------------------------------------- */
footer {
  border-top: 1px solid #ccc;
}
#copyright {
  padding-top: 20px;
  text-align: center;
  color: #999;
}
```

フッターに指定するCSSのソースコード

日本のサンプルの
リーディング・カンパニーを
目指します。

サンプルだからこその"カタチ"がある

わかりやすいサンプルはどうあるべきなのか？　その答えを知っているかどうかでサンプルの出来映えは決まってきます。これはサンプルのテキストです。これはサンプルのテキストです。これはサンプルのテキストです。これはサンプルのテキストです。これはサンプルのテキストです。

これはサンプルのテキストです。これはサンプルのテキストです。これはサンプルのテキストです。これはサンプルのテキストです。これはサンプルのテキストです。

最高のサンプルを驚きのプライスで！



これはサンプルのサイドバーのテキストです。これはサンプルのサイドバーのテキストです。これはサンプルのサイドバーのテキストです。

CM「サンプルプルルン♪」メイキング公開



これはサンプルのサイドバーのテキストです。これはサンプルのサイドバーのテキストです。これはサンプルのサイドバーのテキストです。

一瞬でわかるサンプル

これはサンプルのフッターのテキストです。これはサンプルのフッターのテキストです。これはサンプルのフッターのテキストです。これはサンプルのフッターのテキストです。

見えない部分へのこだわり

これはサンプルのフッターのテキストです。これはサンプルのフッターのテキストです。これはサンプルのフッターのテキストです。これはサンプルのフッターのテキストです。

社会活動・環境活動

これはサンプルのフッターのテキストです。これはサンプルのフッターのテキストです。これはサンプルのフッターのテキストです。これはサンプルのフッターのテキストです。

Copyright © 2012 sample site. All rights reserved.

CSS指定後のフッターの表示。これで一応の完成

■ text-alignプロパティ ………………… p.096へ

PRACTICE 12-6

IE6対応へのヒント

難易度：★★☆☆☆

サンプルページは完成しましたが、HTML5を使う場合には、古いInternet Explorerでの表示に注意する必要があります。ここでは「html5shiv.js」を使って、簡易的に見た目を整える方法を紹介します。

1 さて、これでサンプルページは完成したのですが、もしこれを古いInternet Explorerにも対応させようとすると、ここからさらに長い道のりがはじまってしまいます。参考までに、ここで作成したサンプルページが、Internet Explorer 6だとどのように表示されるのかを紹介しておきましょう。



完成したサンプルページをInternet Explorer 6で表示させたところ

2 CSSが適用されている部分もありますが、まったく適用されていない部分もあります。このようになっている最大の原因は、Chapter 7でも解説したように、古いInternet ExplorerがHTML5の新要素を認識できないことにあります。したがって、コラム「HTML5の新要素を古いIEに認識させる方法(p.129)」で紹介した方法を使えば、表示結果はずいぶんと改善されます。

試しに、「http://code.google.com/p/html5shiv/」で「html5shiv.js」をダウンロード※17して、サンプルページに読み込ませた状態が次のスクリーンショットです(そのほかに、CSSのセレクタとして使用している`nth-child()`を普通のidを対象とした指定に変更しています)。

読み込ませたサンプルファイル

```
sample/chapter-12/practice-12-6/index.html
sample/chapter-12/practice-12-6/styles.css
```



「html5shiv.js」を読み込ませ、セレクタ`nth-child()`を使わないように変更した状態

CHAPTER 12

※17：本書のサンプルファイルには「html5shiv.js」は含まれていません。上のソースコードで実際に試してみる場合には、「http://code.google.com/p/html5shiv/」から最新版の「html5shiv.js」をダウンロードしてサンプルと同じフォルダに入れてください。

3 これだけでも、表示結果はずいぶんと良くなっています。しかし、細かい部分まで見ていくと、修正すべき部分はまだまだ多くあります。たとえば、Internet Explorer 6は透過PNGに対応していません。透過PNGを表示させると、本来透過すべきところが少し青っぽいグレーになって表示されるのです。この問題は、画像を透過しないものに変更するか、透過GIFに変更するなどで対処できます。

それ以外の細かい部分については、Internet Explorerの独自拡張による条件分岐コメントやCSSハックなどというものを使うことで、対処することが可能です(インターネットで検索すると驚くほど多くの対応策が出てきます)。しかし、古いInternet Explorerに対応させるということはそれだけで多くの手間がかかるだけでなく、多くの新機能が使えなくなるということでもあります(つまり設計から変える必要が出てくる場合もあります)。そのため、最近ではInternet Explorer 6に対応させたWebページを制作する場合には別料金が必要となる制作会社も増えてきていますし、対応ブラウザからあえてInternet Explorer 6を外してリリースするWebサービスも多くなっています。

APPENDIX

巻末資料

本書の本文では、未確定の仕様を詳細に解説することは避け、主要な要素・属性のみをピックアップして解説しました。Appendixでは、参考資料として2012年7月現在のHTML5の仕様書で定義されている全要素を掲載し、それがどのカテゴリーに該当するのか、どこに配置できるのか、内容として何を入れることができるのか、といった情報を一覧で示しておきます。

APPENDIX 1

HTML5の要素の分類

Chapter 5では、HTML5の要素のカテゴリーについて説明しました。ここでは、HTML5の全要素を表で掲載し、どの要素がどのカテゴリーに分類されているのかを示します(背景が赤くなっている要素がそのカテゴリーに該当する要素です)。

自由配置コンテンツ(Flow content)

a	command	html	optgroup	sub
abbr	datalist	i	option	summary
address	dd	iframe	output	sup
area	del	img	p	table
article	details	input	param	tbody
aside	dfn	ins	pre	td
audio	div	kbd	progress	textarea
b	dl	keygen	q	tfoot
base	dt	label	rp	th
bdi	em	legend	rt	thead
bdo	embed	li	ruby	time
blockquote	fieldset	link	s	title
body	figcaption	map	samp	tr
br	figure	mark	script	track
button	footer	menu	section	u
canvas	form	meta	select	ul
caption	h1～h6	meter	small	var
cite	head	nav	source	video
code	header	noscript	span	wbr
col	hgroup	object	strong	テキスト
colgroup	hr	ol	style	

※area要素はmap要素に含まれている場合のみ該当
※style要素はscope属性が指定されている場合のみ該当

セクションコンテンツ(Sectioning content)

a	command	html	optgroup	sub
abbr	datalist	i	option	summary
address	dd	iframe	output	sup
area	del	img	p	table
article	details	input	param	tbody
aside	dfn	ins	pre	td
audio	div	kbd	progress	textarea
b	dl	keygen	q	tfoot
base	dt	label	rp	th
bdi	em	legend	rt	thead
bdo	embed	li	ruby	time
blockquote	fieldset	link	s	title
body	figcaption	map	samp	tr
br	figure	mark	script	track
button	footer	menu	section	u
canvas	form	meta	select	ul
caption	h1〜h6	meter	small	var
cite	head	nav	source	video
code	header	noscript	span	wbr
col	hgroup	object	strong	テキスト
colgroup	hr	ol	style	

見出しコンテンツ(Heading content)

a	command	html	optgroup	sub
abbr	datalist	i	option	summary
address	dd	iframe	output	sup
area	del	img	p	table
article	details	input	param	tbody
aside	dfn	ins	pre	td
audio	div	kbd	progress	textarea
b	dl	keygen	q	tfoot
base	dt	label	rp	th
bdi	em	legend	rt	thead
bdo	embed	li	ruby	time
blockquote	fieldset	link	s	title
body	figcaption	map	samp	tr
br	figure	mark	script	track
button	footer	menu	section	u
canvas	form	meta	select	ul
caption	h1〜h6	meter	small	var
cite	head	nav	source	video
code	header	noscript	span	wbr
col	hgroup	object	strong	テキスト
colgroup	hr	ol	style	

文章内コンテンツ（Phrasing content）

a	command	html	optgroup	sub
abbr	datalist	i	option	summary
address	dd	iframe	output	sup
area	del	img	p	table
article	details	input	param	tbody
aside	dfn	ins	pre	td
audio	div	kbd	progress	textarea
b	dl	keygen	q	tfoot
base	dt	label	rp	th
bdi	em	legend	rt	thead
bdo	embed	li	ruby	time
blockquote	fieldset	link	s	title
body	figcaption	map	samp	tr
br	figure	mark	script	track
button	footer	menu	section	u
canvas	form	meta	select	ul
caption	h1～h6	meter	small	var
cite	head	nav	source	video
code	header	noscript	span	wbr
col	hgroup	object	strong	テキスト
colgroup	hr	ol	style	

※a要素・del要素・ins要素・map要素は文章内コンテンツだけを含んでいる場合のみ該当

※area要素はmap要素に含まれている場合のみ該当

組み込みコンテンツ（Embedded content）

a	command	html	optgroup	sub
abbr	datalist	i	option	summary
address	dd	iframe	output	sup
area	del	img	p	table
article	details	input	param	tbody
aside	dfn	ins	pre	td
audio	div	kbd	progress	textarea
b	dl	keygen	q	tfoot
base	dt	label	rp	th
bdi	em	legend	rt	thead
bdo	embed	li	ruby	time
blockquote	fieldset	link	s	title
body	figcaption	map	samp	tr
br	figure	mark	script	track
button	footer	menu	section	u
canvas	form	meta	select	ul
caption	h1～h6	meter	small	var
cite	head	nav	source	video
code	header	noscript	span	wbr
col	hgroup	object	strong	テキスト
colgroup	hr	ol	style	

メタデータコンテンツ（Metadata content）

a	command	html	optgroup	sub
abbr	datalist	i	option	summary
address	dd	iframe	output	sup
area	del	img	p	table
article	details	input	param	tbody
aside	dfn	ins	pre	td
audio	div	kbd	progress	textarea
b	dl	keygen	q	tfoot
base	dt	label	rp	th
bdi	em	legend	rt	thead
bdo	embed	li	ruby	time
blockquote	fieldset	link	s	title
body	figcaption	map	samp	tr
br	figure	mark	script	track
button	footer	menu	section	u
canvas	form	meta	select	ul
caption	h1～h6	meter	small	var
cite	head	nav	source	video
code	header	noscript	span	wbr
col	hgroup	object	strong	テキスト
colgroup	hr	ol	style	

インタラクティブコンテンツ（Interactive content）

a	command	html	optgroup	sub
abbr	datalist	i	option	summary
address	dd	iframe	output	sup
area	del	img	p	table
article	details	input	param	tbody
aside	dfn	ins	pre	td
audio	div	kbd	progress	textarea
b	dl	keygen	q	tfoot
base	dt	label	rp	th
bdi	em	legend	rt	thead
bdo	embed	li	ruby	time
blockquote	fieldset	link	s	title
body	figcaption	map	samp	tr
br	figure	mark	script	track
button	footer	menu	section	u
canvas	form	meta	select	ul
caption	h1～h6	meter	small	var
cite	head	nav	source	video
code	header	noscript	span	wbr
col	hgroup	object	strong	テキスト
colgroup	hr	ol	style	

※audio要素とvideo要素はcontrols属性が指定されている場合のみ該当

※img要素とobject要素はusemap属性が指定されている場合のみ該当

※input要素はtype属性の値が「hidden」以外の場合のみ該当

※menu要素はtype属性の値が「toolbar」の場合のみ該当

APPENDIX 2

HTML5の要素の配置のルール

HTML5の各要素について「その要素はどこに配置できるのか」「その要素の子要素として直接入れられる要素はどれか」という情報を表にまとめました。必要に応じて参照してください。

HTML5の要素の配置のルール

要素名	配置できる場所	子要素として直接入れられる要素
a	自由配置コンテンツが配置できる場所。内容が文章内コンテンツのみの場合は、文章内コンテンツが配置できる場所。	親要素に入れられる要素と同じ(ただし内部にインタラクティブコンテンツを含むことはできない)
abbr	文章内コンテンツが配置できる場所	文章内コンテンツ
address	自由配置コンテンツが配置できる場所	自由配置コンテンツ(ただし内部に見出しコンテンツ・セクションコンテンツ・header要素・footer要素・address要素を含むことはできない)
area	map要素内で文章内コンテンツが配置できる場所	なし
article	自由配置コンテンツが配置できる場所	自由配置コンテンツ
aside	自由配置コンテンツが配置できる場所	自由配置コンテンツ
audio	組み込みコンテンツが配置できる場所	src属性が指定されている場合は、0個以上のtrack要素に続けて親要素に入れられる要素と同じ要素。src属性が指定されていない場合は、はじめに0個以上のsource要素を配置し、次に0個以上のtrack要素、親要素に入れられる要素。いずれの場合も、内部にaudio要素とvideo要素は含むことができない
b	文章内コンテンツが配置できる場所	文章内コンテンツ
base	head要素内(ただし複数は配置できない)	なし
bdi	文章内コンテンツが配置できる場所	文章内コンテンツ
bdo	文章内コンテンツが配置できる場所	文章内コンテンツ
blockquote	自由配置コンテンツが配置できる場所	自由配置コンテンツ
body	html要素内に2つ目の子要素として配置	自由配置コンテンツ
br	文章内コンテンツが配置できる場所	なし
button	文章内コンテンツが配置できる場所	文章内コンテンツ(ただし内部にインタラクティブコンテンツを含むことはできない)

要素名	配置できる場所	子要素として直接入れられる要素
canvas	組み込みコンテンツが配置できる場所	親要素に入れられる要素と同じ
caption	table要素の最初の子要素として配置	自由配置コンテンツ(ただし内部にtable要素を含むことはできない)
cite	文章内コンテンツが配置できる場所	文章内コンテンツ
code	文章内コンテンツが配置できる場所	文章内コンテンツ
col	sapn属性の指定されていないcolgroup要素の子要素として配置	なし
colgroup	table要素の子要素として配置(ただしcaption要素よりも後で、thead要素・tbody要素・tfoot要素・tr要素よりも前に配置)	span属性が指定されている場合は内容は空／span属性が指定されていない場合は0個以上のcol要素
command	メタデータコンテンツまたは文章内コンテンツが配置できる場所	なし
datalist	文章内コンテンツが配置できる場所	文章内コンテンツまたはoption要素を0個以上
dd	dl要素内で、dt要素またはdd要素の後	自由配置コンテンツ
del	自由配置コンテンツが配置できる場所。内容が文章内コンテンツのみの場合は、文章内コンテンツが配置できる場所	親要素に入れられる要素と同じ
details	自由配置コンテンツが配置できる場所	summary要素を1つ、その後に自由配置コンテンツ
dfn	文章内コンテンツが配置できる場所	文章内コンテンツ(ただし内部にdfn要素を含むことはできない)
div	自由配置コンテンツが配置できる場所	自由配置コンテンツ
dl	自由配置コンテンツが配置できる場所	1つ以上のdt要素に続く1つ以上のdd要素のグループを0個以上
dt	dl要素内で、dd要素またはdt要素の前	文章内コンテンツ(ただし内部にheader要素・footer要素・セクションコンテンツ・見出しコンテンツを含むことはできない)
em	文章内コンテンツが配置できる場所	文章内コンテンツ
embed	組み込みコンテンツが配置できる場所	なし
fieldset	自由配置コンテンツが配置できる場所	必要に応じて最初にlegend要素を1つ、その後に自由配置コンテンツ
figcaption	figure要素の最初または最後の子要素として配置	自由配置コンテンツ
figure	自由配置コンテンツが配置できる場所	1つのfigcaption要素に続けて自由配置コンテンツを配置／自由配置コンテンツの最後にfigcaption要素を1つ配置／自由配置コンテンツ

要素名	配置できる場所	子要素として直接入れられる要素
footer	自由配置コンテンツが配置できる場所	自由配置コンテンツ(ただし内部にheader要素・footer要素を含むことはできない)
form	自由配置コンテンツが配置できる場所	自由配置コンテンツ(ただし内部にform要素を含むことはできない)
h1〜h6	自由配置コンテンツが配置できる場所／hgroup要素の子要素として配置	文章内コンテンツ
head	html要素の最初の子要素として配置	1つ以上のメタデータコンテンツ(そのうち1つはtitle要素)。iframeの内容となる文書などの場合は0個以上ののメタデータコンテンツ
header	自由配置コンテンツが配置できる場所	自由配置コンテンツ(ただし内部にheader要素・footer要素を含むことはできない)
hgroup	自由配置コンテンツが配置できる場所	h1〜h6要素を1つ以上
hr	自由配置コンテンツが配置できる場所	なし
html	すべての要素を含むルート要素として配置	head要素とbody要素を順に1つずつ
i	文章内コンテンツが配置できる場所	文章内コンテンツ
iframe	組み込みコンテンツが配置できる場所	テキスト
img	組み込みコンテンツが配置できる場所	なし
input	文章内コンテンツが配置できる場所	なし
ins	自由配置コンテンツが配置できる場所。内容が文章内コンテンツのみの場合は、文章内コンテンツが配置できる場所	親要素に入れられる要素と同じ
kbd	文章内コンテンツが配置できる場所	文章内コンテンツ
keygen	文章内コンテンツが配置できる場所	なし
label	文章内コンテンツが配置できる場所	文章内コンテンツ(ただし内部にlabel要素を含むことはできない。また、label要素と関連付けないbutton要素・input要素・keygen要素・meter要素・output要素・progress要素・select要素・textarea要素は内部に含むことはできない)
legend	fieldset要素の最初の子要素として配置	文章内コンテンツ
li	ul要素内／ol要素内／menu要素内	自由配置コンテンツ
link	メタデータコンテンツが配置できる場所／head要素の子要素であるnoscript要素内	なし
map	自由配置コンテンツが配置できる場所。内容が文章内コンテンツのみの場合は、文章内コンテンツが配置できる場所	親要素に入れられる要素と同じ
mark	文章内コンテンツが配置できる場所	文章内コンテンツ

要素名	配置できる場所	子要素として直接入れられる要素
menu	自由配置コンテンツが配置できる場所	li要素を0個以上。または自由配置コンテンツ
meta	http-equiv属性を指定している場合はhead要素内、またはhead要素の子要素であるnoscript要素内（ただし文字コードの指定はnoscript要素内は不可）／name属性が指定されている場合はメタデータコンテンツが配置できる場所	なし
meter	文章内コンテンツが配置できる場所	文章内コンテンツ（ただし内部にmeter要素を含むことはできない）
nav	自由配置コンテンツが配置できる場所	自由配置コンテンツ
noscript	head要素内、または文章内コンテンツが配置できる場所（ただし内部にnoscript要素を含むことはできない）	head要素内の場合はlink要素・style要素・meta要素を順不同で任意の数／head要素外の場合は親要素に入れられる要素と同じ（ただし内部にnoscript要素を含むことはできない）
object	組み込みコンテンツが配置できる場所	0個以上のparam要素に続けて、親要素に入れられる要素と同じ要素
ol	自由配置コンテンツが配置できる場所	0個以上のli要素
optgroup	select要素の子要素として配置	0個以上のoption要素
option	select要素・datalist要素・optgroup要素の子要素として配置	テキスト
output	文章内コンテンツが配置できる場所	文章内コンテンツ
p	自由配置コンテンツが配置できる場所	文章内コンテンツ
param	object要素の子要素として、どの自由配置コンテンツよりも前に配置	なし
pre	自由配置コンテンツが配置できる場所	文章内コンテンツ
progress	文章内コンテンツが配置できる場所	文章内コンテンツ（ただし内部にprogress要素を含むことはできない）
q	文章内コンテンツが配置できる場所	文章内コンテンツ
rp	ruby要素の子要素として、rt要素の直前または直後に配置	文章内コンテンツ
rt	ruby要素の子要素として配置	文章内コンテンツ
ruby	文章内コンテンツが配置できる場所	文章内コンテンツに続くrt要素、または文章内コンテンツに続くrp要素・rt要素・rp要素を1つ以上
s	文章内コンテンツが配置できる場所	文章内コンテンツ
samp	文章内コンテンツが配置できる場所	文章内コンテンツ

要素名	配置できる場所	子要素として直接入れられる要素
script	メタデータコンテンツまたは文章内コンテンツが配置できる場所	スクリプトのソースコード。src属性がある場合は内容は空またはコメントによる説明のみ
section	自由配置コンテンツが配置できる場所	自由配置コンテンツ
select	文章内コンテンツが配置できる場所	option要素またはoptgroup要素を0個以上
small	文章内コンテンツが配置できる場所	文章内コンテンツ
source	audio要素またはvideo要素の子要素として、他の自由配置コンテンツまたはtrack要素よりも前に配置	なし
span	文章内コンテンツが配置できる場所	文章内コンテンツ
strong	文章内コンテンツが配置できる場所	文章内コンテンツ
style	scoped属性が指定されていない場合はメタデータコンテンツが配置できる場所、またはhead要素の子要素であるnoscript要素内／scoped属性が指定されている場合は自由配置コンテンツが配置できる場所（ただし他の自由配置コンテンツよりも前）	スタイルシートのソースコード
sub	文章内コンテンツが配置できる場所	文章内コンテンツ
summary	details要素の最初の子要素として配置	文章内コンテンツ
sup	文章内コンテンツが配置できる場所	文章内コンテンツ
table	自由配置コンテンツが配置できる場所	次の順に配置：0個か1個のcaption要素／0個以上のcolgroup要素／0個か1個のthead要素／0個か1個のtfoot要素／0個以上のtbody要素または1個以上のtr要素／0個か1個のtfoot要素（ただし、tfoot要素はtable要素内で計1つのみ配置可）
tbody	table要素の子要素としてcaption要素・colgroup要素・thead要素よりも後に配置（ただしtable要素の直接の子要素であるtr要素がない場合のみ）	0個以上のtr要素
td	tr要素の子要素として配置	自由配置コンテンツ
textarea	文章内コンテンツが配置できる場所	テキスト
tfoot	table要素の子要素として、caption要素・colgroup要素・thead要素の後で、tbody要素とtr要素よりも前に配置。または、table要素の子要素として、caption要素・colgroup要素・thead要素・tbody要素・tr要素のあとに配置。ただし、同じtable要素内に別のfooter要素は配置できない	0個以上のtr要素

要素名	配置できる場所	子要素として直接入れられる要素
th	tr要素の子要素として配置	文章内コンテンツ(ただし内部にheader要素・footer要素・セクションコンテンツ・見出しコンテンツを含むことはできない)
thead	table要素の子要素として配置(ただしcaption要素・colgroup要素よりも後で、tbody要素・tfoot要素・tr要素よりも前に配置。同じtable要素内に複数のthead要素は配置できない)	0個以上のtr要素
time	文章内コンテンツが配置できる場所	文章内コンテンツ
title	head要素内に配置(ただし複数は配置できない)	テキスト
tr	thead要素・tbody要素・tfoot要素の子要素として配置／table要素の子要素としてcaption要素・colgroup要素・thead要素よりも後に配置(ただしtbody要素がない場合のみ)	0個以上のtd要素またはth要素
track	audio要素またはvideo要素の子要素として、他の自由配置コンテンツよりも前に配置	なし
u	文章内コンテンツが配置できる場所	文章内コンテンツ
ul	自由配置コンテンツが配置できる場所	li要素を0個以上
var	文章内コンテンツが配置できる場所	文章内コンテンツ
video	組み込みコンテンツが配置できる場所	src属性が指定されている場合は、0個以上のtrack要素に続けて親要素に入れられる要素と同じ要素。src属性が指定されていない場合は、はじめに0個以上のsource要素を配置し、次に0個以上のtrack要素、親要素に入れられる要素。いずれの場合も、内部にaudio要素またはvideo要素を含むことはできない
wbr	文章内コンテンツが配置できる場所	なし

INDEX

キーワード

記号・英数字

日本語

HTML

要素と属性

INDEX

一般的に使用される属性

CSS

ルール

セレクタ

プロパティと値

INDEX

INDEX

著者プロフィール

■**大藤 幹**（おおふじ みき）

札幌在住。大学卒業後、複数のソフトハウスに勤務し、CADアプリケーション・航空関連システム・医療関連システム・マルチメディアタイトルなどの開発に携わる。1996年よりWebの基本技術に関する書籍の執筆を開始し、2000年に独立。その後、ウェブコンテンツJIS（JIS X 8341-3）ワーキング・グループ主査、情報通信アクセス協議会・ウェブアクセシビリティ作業部会委員などを務める。現在の主な業務は、Webデザインに関連する書籍の執筆のほか、全国各地でのセミナー講師など。
著書は『基本からしっかりわかる Movable Type 5 カスタマイズブック』『基本からしっかりわかる WordPress 3.x カスタマイズブック』『Webプロフェッショナルのための黄金則 XHTML+CSS虎の巻』『世界の「最先端」事例に学ぶ CSSベスト・プラクティス』『世界の「最先端」事例に学ぶ CSSプロフェッショナル・スタイル』（以上、マイナビ）、『詳解 HTML&XHTML&CSS辞典』『Pocket 詳解 HTML5&CSS3辞典』（秀和システム）、『10日でおぼえる CSS/CSS3入門教室』（翔泳社）、『XHTML+CSS 超高速コーディング術』（ソシム）など多数。

STAFF

カバーイラスト：2g（http://twograms.jimdo.com）
ブックデザイン：三宮 暁子
写真提供（p.160のplane.png）：山﨑 英人
DTP：ayumu print
担当：伊佐 知子

よくわかるHTML5（エイチティーエムエル ファイブ）＋CSS3（シーエスエス スリー）の教科書（キョウカショ）

2012年7月31日　初版第1刷発行

著者　大藤 幹
発行者　中川 信行
発行所　株式会社マイナビ
〒100-0003　東京都千代田区一ツ橋1-1-1　パレスサイドビル
TEL：048-485-2383（注文専用ダイヤル）
TEL：03-6267-4477（販売）
TEL：03-6267-4432（編集）
E-Mail：pc-books@mynavi.jp
URL：http://book.mynavi.jp
印刷・製本　株式会社ルナテック

ISBN 978-4-8399-4348-6

関連書籍

よくわかるPHPの教科書

たにぐち まこと［著］

ISBN978-4-8399-3314-2

いまさらだけど、PHPを勉強したい。もう1回、
PHPをしっかり理解したい。そんな人にぴったりの入門書です。
平易な言葉で分かりやすく書かれているので、
プログラミングの用語やコードに拒否感がある人でも、
すいすい読み進めることができます。

よくわかるiPhoneアプリ開発の教科書 iOS 5&Xcode 4.2 対応版

森巧尚［著］

ISBN978-4-8399-4173-4

好評の『よくわかるiPhoneアプリ開発の教科書』を、
OSの最新版iOS 5と最新開発環境Xcode 4.2に対応させて
全面改訂。イラスト図解＋サンプル制作で、
基本からしっかりマスターできる、iPhoneアプリ開発の入門書です。

よくわかるAndroidアプリ開発の教科書 Android SDK 2.3 対応

高見知英、松田幸一、椎木啓祐、嶋崎聡［著］

ISBN978-4-8399-3849-9

Androidアプリケーション開発のための、本格派の入門書。
Androidの概要、特徴から、開発環境の構築方法、
制作の基本と流れ、他の環境と比べて注意するべき点、
Androidマーケットへのアップまで。

よくわかるJavaScriptの教科書

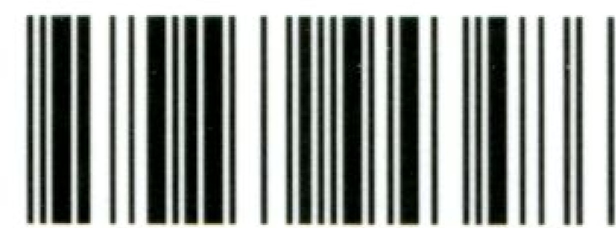
たにぐち まこと［著］

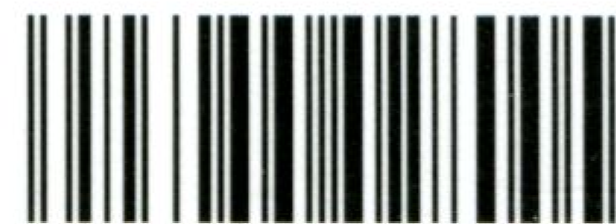
ISBN978-4-8399-4187-1

JavaScriptの基本から、jQuery、jQueryMobileまで1冊で。
プログラマでない人でも読みやすいように、やさしい言葉を
使いながら、1つひとつ丁寧に説明している本ですので、
途中で迷うことなく学習を進めることができます。

ISBN978-4-8399-4348-6
C3055 ¥2800E

定価：本体 2,800円 ＋税





よくわかるHTML5＋CSS3の教科書



大藤 幹［著］

マイナビ